STANFORD ARTIFICIAL INTELLIGENCE LABORATORY October 18, 1968
OPERATING NOTE NO. 1

ALVINE

by John Allen

ABSTRACT: A new LISP editor, Alvine is now available. Significant improvements have been made in the command structure and speed of Alvine. The major addition to Alvine is a pointer which can be moved through the editor's string; the editing features affect only the area to the right of this pointer. Once can insert and delete arbitrary character strings; and file and defile these strings on various I-O devices.

The data for Alvine are aribtrary strings of LISP atoms, numbers parens and dots. The Alvine commands are designed to edit these strings into LISP S-expressions with a minimal amount of fuss. The editor is initially equipped to handle the special indicators, FEXPR, EXPR and VALUE. This list may be ammended by the programmer (see the description of "G").

Description of the Command Structure

Each command to Alvine consists of a single character followed by a string of arguments. These commands modify the text string presently occupying Alvine's buffer. When text is introduced the Alvine a pointer is attached preceeding the first object in the buffer. Alvine's commands allow the user to move this pointer through the buffer. Alvine's text modifying commands only affect the string to the right of this pointer.

The modification commands include insertion and deletion of material, "pointer string" refers to the string to the right of the pointer.

COMMAND	MEANING	DESCRIPTION
A	All	Print the buffer string. No attempt is made to make the output pretty.
B	Balanced?	Examines the number of parens in the buffer string. Returns the count of left and right parens if unbalanced; otherwise replies "BAL".
nC	Count	For readability, the commands "D", "M", ">", "<", "S", and "W", will

		print an initial segment of the pointer buffer. "nC" sets the length of this printing segment to n objects.
nD	Delete	Delete the first n objects to the right of the pointer. If n is omitted, 1 is assumed.
E	Expunge	Expunge the first S-expression in the pointer buffer.
F x y: z	File	Files the material referred to by "x" on device "y" using "z" as a file name. If "x" is a list then each element of "x" is filed under "z"; if "x" is an atom then "x" is assumed to be "SETQ"-ed to a list of names to be filed. "F" also manufactures line numbers, giving an EDITOR-compatible text.
Gx	Get	G will move the S-expression with name "x" into the Alvine buffer and initialize the pointer to the left-hand end of the buffer. If the indicator associated with "x" is "VALUE" then Alvine sees: (SETQ x (QUOTE . . .)) otherwise (DEFPROP x ( ) indicator). Alvine looks for the indicators on the list named "%%%L", which is initialized to "(FEXPR EXPR VALUE)". %%%L may be amended as needed. G also knows about traced functions and will edit them properly.
I	Insert	Insert comes in two flavors: 1. IX: insert "x" immediately to the right of the pointer. 2. Ixy: insert "y" after the first occurrence (in the pointer string) of the string "x". "x" may be a complete string or described by ellipsis as "w ... z". If "x" is % then "y" is introduced to the editor as the current string.
M	Match	Move the pointer one s-expression to the right of the current pointer position; if there is no such s-expression the pointer is not moved and the bell is sounded.

P	Put	Returns the editor string to LISP format through EVAL. Thus for example DEFPROP and SETQ are handled by the editor.
Rxy	Replace	Replace the first occurrence of "x" by "y". As with "I", "x" may be described elliptically; and if "y" is %, the first occurrence of "x" is deleted.
nSx$	Search	Search for the n^{th} occurrence of the string "x" (in the pointer string). If found, the pointer is moved to the beginning of the string following that occurrence. If less than n occurrences are located, the pointer is positioned after the last such occurrence. If none are found the pointer is not moved. If "x" is not given, i.e., "nS$", then the last given search-string is used.
V	Vomit	Print the first balanced paren section to the right of the pointer in pseudo GRINDEF format.
W	Where?	Prints the beginning of the pointer string.
n > and n <		These commands are dual; they move the string pointer "n" objects to the right or left respectively. If "n" is such that either the left or right end of the string would be exceeded the pointer is set to that extreme, and "bell" is typed. To reset to the extreme left of the string "Ø<" may be used.
↑		This command returns control to LISP. Alvine's buffer is left intact, and returning to Alvine; the user will find the pointer at the left hand end of the old string.
Bell		Bell may be used during any command to return control to Alvine's command listen-loop.

SAILON No. 1

AN EXAMPLE OF ALVINE

Note: 1. All typeout is underlined.

2. Bell, space and alt-mode are represented by ⊓, ␣ and $ respectively.

```
↑C↵
.
.R LISP 12↵
LISP 22-Aug-68
ALLOC? N␣↵
T␣↵
T ↵
( *R  ALVINE $↵
*
I % $(DEFPROP TEST(LAMBDA $; the string bounded by "$" is introduced
                                    to ALVINE
*
A␣
(DEFPROP TEST(LAMBDA ; print the entire ALVINE buffer
*
B␣
2LPS
Ø RPS
*
I LAMBDA $ (X) (CAR Y) EXPR) $; append the string bounded by "$" to
                                    the buffer
*
B␣
4LPS
3 RPS
*
I CAR Y $)$ ; add the deficient right paren
*
                                    ; the following commands would also
                                      have the same effect:
                                    ; 1.  "11<", "I $)$"
                                    ; 2.  "SY$", I $)$
B␣
BAL ;
*
V␣
(DEFPROP TEST (LAMBDA (X) (CAR Y))EXPR)
*
P TEST␣; convert ALVINE string to LISP function
*
```

SAILON No. 1

```
↑  ;  exit ALVINE
T␣;  now talking to LISP
T
(TEST (QUOTE(A B)))
Y
UNBOUND VARIABLE-EVAL  ; LOOSE
(ALVINE)  ; reenter ALVINE, "(ED" will also work.
*
W␣
(DEFPROP TEST ; "G" need not be executed since the buffer is always
*                                    left intact.
R X $ Y$
*
P TS∏  ; flush incorrect "put" command by typing bell. (∏)
*
PTEST␣ ; redefine TEST
*
↑
(TEST (QUOTE(A B)))␣ ; try again
A                    ; win
(ED)
*
5>␣
(Y)
*
5C␣ ; change print count
*
M␣
(CAR Y))
*
E␣
*
A␣
(DEFPROP TEST (LAMBDA (Y))EXPR)
*
W␣
)EXPR)
*
0<␣
∏(DEFPROP TEST(
*
R TEST ...) $%$
*
A␣
(DEFPROP)EXPR)  ; same effect by :
                  1.  "SDEFPROP $", "6D"
*
```