OVERVIEW OF A DISPLAY-ORIENTED EDITOR FOR INTERLISP

David R. Barstow

Schlumberger-Doll Research
Ridgefield, Connecticut

ABSTRACT

DED is a display-oriented editor that was designed to add the power and convenience of display terminals to INTERLISPs teletype-oriented structure editor. DED divides the display screen into a Prettypnnt Region and an Interaction Region. The Prettypnnt Region gives a prettyprinted view of the structure being edited; the Interaction Region contains the interaction between the user and INTERLISPs standard editor. DEDs prettyprinter allows ellision, and the user may zoom in or out to see the expression being edited with more or less detail. There arc several arrow keys which allow the user to change quite easily the locus of attention in certain structural ways, as well as a menu-like facility for common command sequences. Together, these features provide a display-facility that considerably augments INTERLISPs otherwise quite sophisticated user interface.

[Note: an extended version of this paper, a user's manual, and source files are available from the author.)

I INTRODUCTION

Although programs are written in a programming language, the human process *of* programming often occurs within a programming environment -- a set of software tools that assist in different aspects of the programming process [1]. A major feature of the INTERLISP [2] environment is a structure-oriented editor whose commands operate on structural objects (e.g., atoms, S-expressions) rather than on textual objects (e.g., characters, lines). These commands deal with and modify in-core S-expressions rather than external textual representations INTERLISP's editor has evolved into a highly sophisticated tool. It's sophistication, however, is marred by one major drawback: it is built around a teletype-style interaction. If you wish a visual presentation of the expression being edited, you must explicitly request that it be typed out on the terminal. While this made sense when INTERLISP was originally developed (the late 1960's when display terminals were rare), it is now a hindrance, since display technologies can add considerable power and comfort to a human interface. A display-oriented editor (called DEI)) has been built to bring some of this power and comfort to the INTERLISP environment.

DED includes the following features:

- The display screen is divided into two regions: one is for the standard style of interaction with INTERLISP's editor; the other gives a prettyprinted version of the expression being edited.
- The prettyprinter allows ellision: the most detailed parts of the prettyprinted expression are shown in abbreviated form.
- The user may zoom in or out to see the expression with more or less detail.
- There are several "arrow" keys which can be used to move the "cursor" *on* the prettyprinted expression (in addition to INTERLISP's usual attention-changing commands).
- A menu-like facility allows common command sequences to be entered quickly.

Together, these features provide a display facility which considerably augments INTERLISP's editor. Users have found it quite easy to learn and have not felt that it disrupts those aspects of INTERLISP that they already use. Thus, the original goal of DED has been achieved: much of the power of display terminals has been added to INTERLISP without losing the structural orientation of the editor and without forcing radically new patterns of behavior onto users. INTERLISP with DED thus represents one solution to the problem of combining a program's visual representation with structural editing. (MACLISP/EMACS [3], [4] and the Cornell Program Synthesizer [5] represent alternative solutions to this problem.)

II THE USER'S VIEW

The user interface to DED is a character-oriented display terminal with 24 lines of 80 characters each. The 24 lines are broken into two main regions, the Prettyprint Region and the Interaction Region. The Prettyprint Region contains a prettyprinted version of the expression being edited. INTERLISP\s current focus of attention is indicated within the Prettyprint

Region by highlighting the enclosing parentheses. The Interaction Region contains the interaction between the user and INTERLISP's standard editor (i.e., the prompt characters, the user's typed command, and INTERLISP's response). For example, Fig. 1 shows the

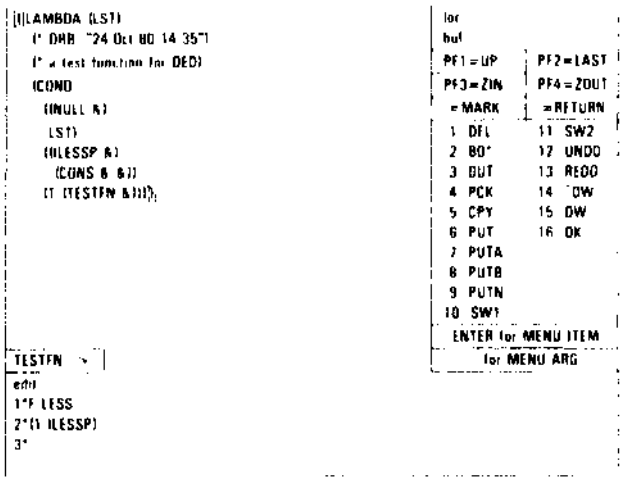

Fig 1 Starting to edit TESTEN

screen when the user is editing the definition of the function TFSTFN. Note that the definition is not printed in full detail; omitted subexrpessions are shown as "&" in the Prettyprint Region.

DED allows the user to zoom in to see specific subexpressions in more detail. In order to indicate the location of the displayed expression within the entire expression, the CARs of the nested expressions surrounding the displayed expression are shown in the line (called the "zoom line") separating the Preltyprint Region and Interaction Region. The zoom out command causes DED to redisplay the expression which was displayed prior to the most recent zoom in command. That is, DED maintains a stack of displayed expressions; zooming in corresponds to a push operation, zooming out corresponds to a pop operation.

DED always ensures that the user's focus of attention is a subexpression of the expression displayed in the Preltyprint Region. Thus, for example, if a search command shifts the focus of attention to an expression not contained in the current expression, DED zooms out until the displayed expression contains the new focus, and then, if necessary, zooms in to an expression such that the new focus is visible despite the ellision of the pretlyprinter.

In order to facilitate moving about within an expression DED includes several single-key commands to change the locus of attention. These include keys to move up and down levels, and right and left within a level. None of these keys ever causes the focus of attention to move outside of the currently displayed expression. An unusual feature of these keys is that some other action is performed if the normal one does not fit the context, For example, if the current focus of attention is on the last element of a list, the right arrow moves up a level before moving lo the next element.

Editing with DED is carried out either by issuing standard INTER LISP commands, or by using a menu-like facility (the upper right corner of Fig. 1) for special structure modification commands (e.g., moving an expression from one place lo another).

III NOTES ON THE IMPLEMENTATION

A. The Display Chain

INTERLISPs editor was originally based on the concept of a "current expression", that part of the structure which is the user's current focus of atlention. To this, DED adds a "display expression", the part of the structure which is displayed at a given time. DED is built so that the current expression is always within (that is, a substructure of) the display expression. In order to keep track of the location of the current expression within the top expression, INTERLISP's editor uses a structure called an "edit chain", which is a list of expressions. The first expression is the current expression, the last expression is the top expression, and each expression on the list is either a member or a tail of the expression following it on the list. Thus, the edit chain provides a stack of nested expressions, leading from the top expression to the current expression. In a similar manner, DED maintains a "display chain," a stack of nested expressions lo be displayed. Expressions are added lo and removed from the display chain as the user zooms in and out. The display chain is always a subsequence of the edit chain.

B. Extending DED to Other Display Devices

DED was built to use a DEC VT100 terminal. There are three features of the VT100 that are crucial to DED's success: arbitrarily many portions of the screen may be highlighted (using reverse video); the current cursor location may be requested from the terminal; and the scrolling region may be restricted. In an attempt to gain device independence, all of DED's interactions with the display are controlled by a table describing the appropriate protocols for different features This technique permitted DED to be extended to a Zenith (Heathkit) terminal quite easily.

After DED was finished, a small experiment was performed to test the feasibility of extending it to run on a bitmap display (a Xerox Dolphin). This turned out lo be simpler than expected, involving essentially a mapping between character positions and bit positions. More interesting was the attempt to use a mouse (rather than arrow keys) to indicate changes in the focus of attention. Although the experiment has not yet been completed, it has become clear that a pointing device, especially when combined with a sophisticated menu system, was a significant improvement over the VT100 interface.

IV LESSONS AND SURPRISES

Building programming environments is inherently an experimental process -- it is simply impossible to determine betorehanti the appropriate way for an environment to appear to the user and to respond to his/her requests. DED is the second display-editor 1 have built for INTFRFFSP (but the first that I actually use) The following are the major lessons to be learned from the DED experience (They reflect primarily my own experience in using DED, but also the collected experiences of a small but growing user community.)

- Automatic zooming to ensure that the focus of attention is always visible is surprisingly valuable.
- The zoom line as a way of determining the location of the displayed expression within the entire expression is surprisingly unimportant
- The arrow keys, especially with their secondary definitions, are a convenient way to change the focus of attention if no pointing device is available

A final observation is that my editing syle has changed considerably. The preltypnnier with ellipsis provides (for me, at least) a comfortable level at which lo view a form - so comfortable that 1 find myself referring much less frequently to hardcopy listings. Adding display facilities to INTFRFISPs structure editor has oriented my editing style much more toward interactivity than it ever was in the past

ACKNOWLEDGEMENTS

Steve Weslold and Bill van Melle have made many helpful suggestions. Feedback from the INTFRFFSP community at SDR has also been quite valuable

REFERENCES

[1] Barstow, D., Shrobc, H, and Sandewall, F (eds.) *Interactive Programming Environments,* McGraw-Hill, 1981.

[2| Moon, I)., "The MacFisp Reference Manual", MIT AI Fab, 1974.

[3] Stallman, R. "Fmacs, the extensible, customizable, sell-documenting display editor", AI Memo 519a, MIT AI Fab, May 1981, included in *Interactive Programming Environments.*

|4) Teitelbaum, R and Reps, T. "The Cornell Program Synthcsi/er: a syntax-directed programming environment", *Communications of the ACM,* (lo appear); included in *Interactive Programming Environments.*

|5| Teitelman, W. and Masinter, F., "The INTFRLISP Programming Fnvironment", Computer April 1981, included in *Interactive Programming Environments.*