

Emacs

“A Thermonuclear Text Editor”

David Hilley

davidhi@cc.gatech.edu

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332

Emacs

- ➔ What is Emacs?
 - ➔ A text editor
 - ➔ A Lisp interpreter
 - ➔ A bytecode compiler
- ➔ What's this controversy?
 - ➔ Editor wars
 - ➔ Emacs vs. `vi/vim`
 - ➔ GNU Emacs vs. XEmacs

“Emacs is the extensible, customizable, self-documenting real-time display editor.”

– GNU Emacs manual

Emacs History

- ➔ Where did Emacs come from?
 - ➔ Editor **MACroS** for TECO (1976)
 - ➔ EINE (1976) / ZWEI (1978)
 - ➔ Gosling Emacs (1981) / Unipress Emacs (1983)
 - ➔ GNU Emacs (1984)
 - ➔ Lucid Emacs (1992) / XEmacs (1994)

“EMACS is a nice editor too, but because it costs hundreds of dollars, there will always be people who won’t buy it.”

– Bill Joy, creator of `vi` (1984)

Emacs Advantages

- ➔ Completely extensible/programmable
- ➔ Modeless editing
- ➔ Available for many platforms
- ➔ Large support community
- ➔ Many different additional functions (mail client, newsreader, web browser, IDE, AIM client, IRC client, shell, psychotherapist, etc.)

“One of the good things about EMACS, though, is its programmability and the modelessness. Those are two ideas which never occurred to me. ”

– Bill Joy, creator of `vi` (1984)

Emacs Criticisms

- ➔ Slow? Memory Hog?
- ➔ **E**ight **M**egs **A**nd **C**onstantly **S**wapping
- ➔ Not always available by default (`vi` is POSIX)
- ➔ Emacs should be replaced by Escheme, CL, ...
- ➔ **E**scape **M**eta **A**lt **C**ontrol **S**hift
- ➔ “Why do I need Tetris™ in my editor?”

“Shimmer is both a floor wax and a dessert topping!”

– Chevy Chase, SNL

Emacs Specifics

- ➔ Modeless Editing
- ➔ Modifier Rich (Ctrl, Meta, Alt, Super, Hyper)
- ➔ C-h t means ('Control' and 'h', then hit 't').

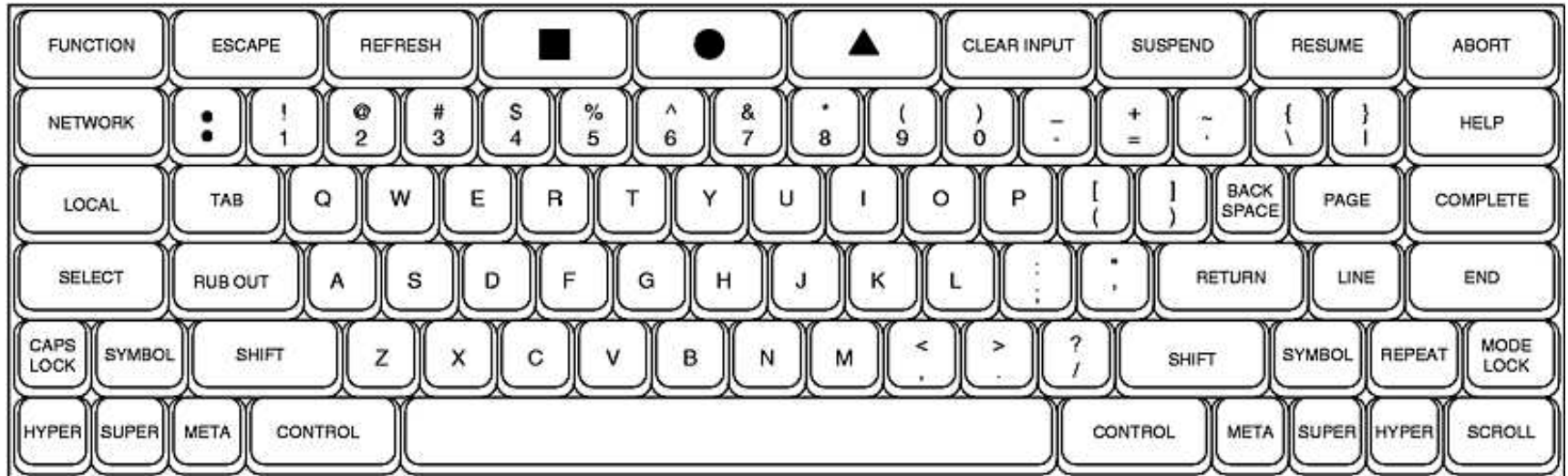


Figure 1: Symbolics Keyboard

Related Memorabilia



Figure 2: Space-Cadet Keyboard

“It remains a monument to design excess.”

– <http://world.std.com/~jdostale/kbd/>

Emacs Terminology

- Region (selection)
- Kill C-k (cut)
- Yank C-y (paste)
- Frame, Window
- Buffer
- Modeline
- Minibuffer

```
File Edit Options Buffers Tools Preview LaTeX Command Help
\item Give a BV"{u}chi automaton  $B$  that accepts the language  $A = (0 + 1)^*\omega$ . \vspace{4pt} \newline It seems to me that this language is the same as the one in Figure 9.7 of the Model Checking book (with  $a$  and  $b$  changed to  $0$  and  $1$ , or course), since the language  $A$  is the set of all infinite strings with a finite number of  $0$  occurrences.
\begin{center}
[height=1.2in]{buchi1}
\end{center}
\item Give languages  $S$  and  $L$  such that  $S$  is safe,  $L$  is live and  $A = S \cap L$ . Give a proof. \newline Let  $S$  be the language  $(0 + 1)^*\omega$ . The closure of  $S$  is  $S$ , so it is a safety property. Let  $L$  be the language where the number of  $0$ 's is finite. The closure of  $L$  is  $\Sigma^*\omega$ , thus it is a liveness property, and  $S \cap L = A$ .
\item Apply the subset construction to  $B$ . What is the resulting automaton and what language does it accept? \vspace{4pt} \newline Since deterministic and non-deterministic BV"{u}chi automata are not equivalent in power, using subset construction changes the language recognized. This automaton recognizes the language
-----F1 final.tex 6:50PM 0.13 (LaTeX Fly)--L37--C2--18%-----
val res' = (map (fn _ => node) block) @ res in
(case block of
[] => addblocks(NONE, [], nnodes, graph,
                defs, uses, ismoves, res')
| (A,LABEL{lab, ...})::_ => let
    val (d_list, u_list) = calcDefsUses(block,
                                        RegSet.empty,
                                        RegSet.empty)
    val defs' = Graph.Table.enter(defs, node, d_list)
    val uses' = Graph.Table.enter(uses, node, u_list)
    val nnodes' = Symbol.enter(nnodes, lab, node)
    val ismoves' = Graph.Table.enter(ismoves, node, false) in
    addblocks(SOME(node), rs, nnodes', graph,
              defs', uses', ismoves', res')
end
| [A.MOVE {dst, src, ...}] => let
    val defs' = Graph.Table.enter(defs, node, [dst])
    val uses' = Graph.Table.enter(uses, node, [src])
    val ismoves' = Graph.Table.enter(ismoves, node, true) in
    addblocks(SOME(node), rs, nnodes', graph,
              defs', uses', ismoves', res')
end
-----F1 make-graph.sal 6:50PM 0.13 (SML/S-1.2)--L72--C0--31%-----
M-x
```

Emacs Modeline

- Isn't Emacs modeless?
- One Major mode (text, directories, C++, etc.)
- Many minor modes (spell checking, CVS, etc.)
- Time, load average
- Character encoding
- Line #, Column #, % of buffer above screen
- Buffer name
- modification status (--/* * or %%/ % *)

```
----:---F1  *scratch*      9:14PM 0.33  (Lisp Interaction)--L1--C0--All-----
```

Emacs Basic Commands

- ➔ C-x C-f Open / Create file
- ➔ C-x C-s Save file
- ➔ C-x C-c Quit
- ➔ C-x k Kill current buffer
- ➔ C-x 2 Split window
- ➔ C-x o Switch (Other) window
- ➔ C-x 1 Unsplit window
- ➔ C-x b Switch buffer
- ➔ C-x C-b Buffer List

Emacs Basic Commands cont.

- ➔ C-_ or C-/ Undo
- ➔ M-x Execute function (M-p and M-n for history)
- ➔ C-g Abort action
- ➔ C-x ESC ESC Last function call

“Emacs outshines all other editing software in approximately the same way that the noonday sun does the stars. It is not just bigger and brighter; it simply makes everything else vanish.”

– Neal Stephenson, sci-fi author

Emacs Help Commands

- ➔ C-h t Emacs Tutorial
- ➔ C-h k What does this key do?
- ➔ C-h f What does this function do?
- ➔ C-h v What is this variable?
- ➔ C-h m What are the keys for this mode?
- ➔ C-h w What keys are bound to this function?
- ➔ C-h a Which commands match this string?
- ➔ C-h i Info pages for everything.
- ➔ C-h C-h Help on help.

Emacs Customization

- ➔ `~/.emacs` file contains your configuration info
- ➔ Emacs Lisp
 - ➔ Lisp dialect
 - ➔ Compiled into bytecode
 - ➔ Dynamic scope
 - ➔ Lexical closures are possible (hack)
 - ➔ Does not optimize tail recursion
- ➔ Simple customizations: `(setq column-number-mode t)`
- ➔ Hooks to add code:
`(add-hook 'text-mode-hook 'flyspell-mode)`

EmacsClient / GnuClient

- ➔ EmacsClient / Server
 - ➔ Included with Emacs 21
 - ➔ Uses running server emacs process to edit
 - ➔ MultiTTY support in CVS Emacs (next version)
 - ➔ Arbitrary elisp execution in client in CVS Emacs
- ➔ GnuClient / Server
 - ➔ Separate package for GNU Emacs
 - ➔ Included with XEmacs
 - ➔ ScreenServer for terminals using GNU Emacs 21
 - ➔ Already supports win32

Emacs Nifty Stuff

- ➔ Eshell
 - ➔ Entirely in Elisp, so platform independent
 - ➔ Features from bash, tsch, zsh, 4nt
 - ➔ Pseudo devices like `/dev/clip`
- ➔ ISwitchB: `(iswitchb-mode 1)`
- ➔ SavePlace
- ➔ Bookmarks
- ➔ AutoCompress: `(auto-compression-mode 1)`
- ➔ RefillMode: `refill-mode`

Emacs Nifty Stuff cont.

- ➔ Minor Modes
 - highline, abbrev-mode, flyspell
 - VC: CVS, SubVersion (included with CVS Emacs)
- ➔ Dictionary Mode
- ➔ TRAMP (included with CVS Emacs), HoboMode
- ➔ Version Controlled Backups
- ➔ Tabbar Mode
- ➔ Dired Mode
- ➔ Diminish Modeline

Emacs Development

- ➔ Lisp, Scheme, C, C++, Java, Awk, Fortran, Icon, Objective-C, Delphi, Pascal, Perl, Tcl, Ada, Modula2, Matlab, Simula, Prolog, Makefile, sh, Asm Mode, ...
- ➔ CEDET-related (Collection of Emacs Development Environment Tools)
 - ➔ SpeedBar
 - ➔ JDEE (Java Development Environment for Emacs)
 - ➔ ECB (Emacs Code Browser)
- ➔ Eldoc mode
- ➔ OO-browser
- ➔ hideshow minor mode (code folding)

Emacs Development cont.

- ➔ CScope
- ➔ Etags / Exuberant Ctags
- ➔ Python mode (in CVS Emacs), SML-mode (with SML dist)
- ➔ Quack mode (for mzscheme based Scheme)
- ➔ GUD (Grand Unified Debugger)
 - Presents unified interface for interactive debugging
 - Works with gdb, dbx, xdb, sdb, perlldb, jdb, and pdb
 - Also, for gdb: gdb-ui/mi (in CVS emacs)
- ➔ SQL Mode, PLSQL Mode, SQLEd Mode

Emacs $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ & Markup

- ➔ $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ mode, $\text{BIB}\text{T}_{\text{E}}\text{X}$ mode
- ➔ $\text{AUC}\text{T}_{\text{E}}\text{X}$
- ➔ preview-latex
- ➔ LilyPond Mode
- ➔ NXML Mode, PSGML Mode
- ➔ HTML Mode, HTML-Helper Mode, CSS Mode

“Though it’s best known as a powerful text editor favored by UNIX developers, Emacs can be used to work with XML in non-UNIX platforms such as Windows, MS-DOS, and MacOS. Emacs works as a full-blown development environment for processing text, writing applications, and, as I’ll discuss, creating structured information like XML and SGML.”

– Brian Gillan, IBM

Emacs Apps

- ➔ Mail Clients
 - ➔ VM
 - ➔ Rmail
 - ➔ Gnus (also newsreader)
- ➔ BBDB (The Insidious Big Brother Database)
- ➔ Web Browsers
 - ➔ w3
 - ➔ emacs-w3m
- ➔ erc (IRC client), tnt (AIM client)
- ➔ The Remembrance Agent

Emacs Resources

- ➔ GNU Emacs: <http://www.gnu.org/software/emacs/>
- ➔ EmacsWiki: <http://www.emacswiki.org>
- ➔ DotEmacs: <http://www.dotemacs.de/>
- ➔ Emacs Timeline/History:
<http://www.jwz.org/doc/emacs-timeline.html>
- ➔ Elisp Introduction:
<http://www.cs.indiana.edu/elisp/elisp-intro.html>
- ➔ Another Elisp Intro:
<http://www.rattlesnake.com/intro/>