The ACL2 Prooftree and Mouse Interface

M. Kaufmann & M. K. Smith

1. Introduction

NOTE: This interface should be considered preliminary, although it has been used successfully at Computational Logic, Inc. It is not part of the ACL2 software in the strictest sense (which is co-authored by Matt Kaufmann and J Moore), but we feel that it will be useful to ACL2 users.

This note describes how to get the ACL2/Emacs prooftree and mouse support. You just need to add a single autoload form to your .emacs file. And then issue the correspoding M-x command.

The prooftree support has been tested in the following Emacs:

Emacs 18
Emacs 19 - with comint and WFS's shell, sshell.el.
Lemacs 19

The menu and mouse support currently works with Emacs 19.

If you don't want to deal with any of this: You probably want to put the following form in your acl2-customization.lisp file.

```
:STOP-PROOF-TREE
```

This will turn off the proof tree printing from ACL2. For documentation in ACL2 do

```
:doc proof-tree
```

To turn proof trees back on use ':START-PROOF-TREE'.
NOTE: If you do ':STOP-PROOF-TREE' in ACL2, then M-x start-proof-tree will not accomplish anything useful in Emacs.

2. LOADING EMACS INTERFACE CODE

2.1 Simplest .emacs Additions

If you want the full interface, put the following in your .emacs file after replacing /slocal/src/acl2/v1-8/ with the full pathname of your acl2-sources/ directory.

```
(setq *acl2-interface-dir*
  "/slocal/src/acl2/v1-8/interface/emacs/")

(autoload 'run-acl2 ;;emacs 19.27 only at this time
  (concat *acl2-interface-dir* "top-start-inferior-acl2")
  "Begin ACL2 in an inferior ACL2 mode buffer."
  t)
```

Then, to get things started in Emacs do 'M-x run-acl2'. Use 'M-x acl2-mode' to get '<acl2-file>.lisp' into the right mode. The commands in the various modes are listed in a later section. But you can see most of them by observing the new pull-down menus and pop-up menu in inferior ACL2 mode and ACL2 mode. The pop-up menu is tied to mouse-3.

If you just want proof trees, use the following after replacing /slocal/src/acl2/v1-8/ with the full pathname of your acl2-sources/ directory.

```
 (setq *acl2-interface-dir*
   "/slocal/src/acl2/v1-8/interface/emacs/")

(autoload 'start-proof-tree
  (concat *acl2-interface-dir* "top-start-shell-acl2")
  "Enable proof tree logging in a prooftree buffer."
  t)
```

2.2 More Control from .emacs: Setting preferences

The alist, *acl2-user-map-interface*, determines what menus you get. If a feature is included after a mode name, then you get it.

```
(defvar *acl2-user-map-interface*
  '((inferior-acl2-mode menu-bar popup-menu keys)
    (acl2-mode          menu-bar popup-menu keys)
    (prooftree-mode     menu-bar popup-menu keys)))
```

If you set the following to T, you will switch to the inferior ACL2 buffer when you send forms, regions, or buffers to it.

```
 (setq *acl2-eval-and-go* nil)
```

If you set the following to NIL you will be queried for their values when you start up a prooftree buffer (via M-x start-proof-tree). These are the defaults you get based on the autoload above.

```
 (setq *acl2-proof-tree-height* 17)
 (setq *checkpoint-recenter-line* 3)
```

3. Commands

Commands are enabled based on the value of the alist, *acl2-user-map-interface*, as described above. There are some conventions that you need to know regarding arguments to mouse commands.

If a menu bar entry is of the form

Print event ...

the "..." indicates that you will be prompted in the minibuffer for an argument.

If a menu bar entry is of the form

Mode >

the ">" indicates a suborninate menu that will pop up if you release on this menu item.

Pop-up menu items indicate whether they take an argument based on a preceding ".". The argument is determined by what you clicked on to bring up the menu. Arguments derived from things that appear in the chronology are somewhat robust. So that if you had a list of events on the screen like:

```
          13  (DEFMACRO TEXT (X) ...)
 L        14  (DEFUN MSG-P (X) ...)
 L        15  (DEFUN MAKE-PACKET (X Y Z) ...)
 L        16  (DEFUN HISTORY-P (L) ...)
          17  (DEFMACRO INFROM (X) ...)
```

to see event 14 you could click right anywhere on that line and select either ". Print Event" or ". Print Command".

3.1 Prooftree Related

M-x start-proof-tree
M-x stop-proof-tree

3.2 Prooftree Mode

3.2-A POPUP MENU

Abort	Abort *inferior-acl2*.
Goto subgoal	Go to clicked on subgoal in *inferior-acl2*.
Resume proof tree	Resume printing proof tree.
Suspend proof tree	Suspend printing proof tree.
Checkpoint/Suspend	Suspend prooftree and go to clicked on checkpoint.
Checkpoint	Go to clicked on checkpoint.
Help	

3.2-B MENU BAR

Prooftree

Checkpoint	Go to next checkpoint
Goto subgoal	That cursor is on.
Checkpoint / Suspend	Go to next checkpoint and suspend proof tree.
Resume proof tree	
Suspend proof tree	
Abort	Abort prooftree. (ACL2 will continue to send prooftrees, it just won’t go the the prooftree buffer.)
Help	

3.2-C KEYS

C-z z	Previous C-z key binding
C-z c	Go to checkpoint
C-z s	Suspend proof tree
C-z r	Resume proof tree
C-z a	Mfm abort secondary buffer
C-z g	Goto subgoal
C-z h	help
C-z ?	help

3.3 ACL2 Mode

ACL2 Mode is like Lisp mode except that the functions that send sexprs to the inferior Lisp process expect an inferior ACL2 process in the *inferior-acl2* buffer.

3.3-A POPUP MENU

Send to ACL2	Send top level form clicked on to ACL2.
Add hint	Add the hint form to the clicked on defun.
Do not induct.	
Do not generalize.	
Do not fertilize.	
Expand	expand form. Requests you mouse it.
Hands off.	
Disable	Disable symbol. Requests you mouse it.
Enable	Enable symbol. Requests you mouse it.
Induct	Induct based on form. Requests you mouse it.
Cases	Perform case split on form.Requests you mouse it.
Go to inferior ACL2	
Verify	Take clicked on form into interactive prover.

3.3-B KEYS

C-x C-e	eval last sexp
C-c C-r	eval region
C-M-x	eval defun
C-c C-e	eval defun
C-c C-z	switch to ACL2
C-c C-l	load file
C-c C-a	show arglist
C-c C-d	describe symbol
C-c C-f	show function documentation
C-c C-v	show variable documentation
C-ce	eval defun and go to ACL2
C-cr	eval region and go to ACL2

3.4 Inferior ACL2 Mode

3.4-A MENU BAR

Events

Recent events	(pbt ’(:here -10))
Print back through ...	(pbt <event>)
Undo	(ubt ’:here)
Oops	(oops)
Undo through ...	(ubt ’<event>)
Undo through ...	(ubt! ’<cd>)
Load file ...	(cl2-load-file)
Disable ...	(in-theory (disable <symbol>))
Enable ...	(in-theory (enable <symbol>))
Verify guards ...	(verify-guards ’<symbol>)

- Verify termination ... (verify-guards ’<symbol>)

- Certify-book ... (certify-book <filename>)
- Include-book ... (include-book <filename>)

- Compound commands
  - Expand compound command ... (puff ’<cd>)
  - Expand compound command! ... (puff* ’<cd>)

- Table
  - Print value ... (table symbol)
  - Clear ... (table <symbol> nil nil :clear
  - Print guard ... (table <symbol> nil nil :guard)

Print

- Event ... (pe ’event)
- Event! ... (pe! ’event)
- Back through ... (pbt ’event)

- Command ... (pc ’<cd>)
- Command block ... (pcb ’<cd>)
- Full Command block ... (pcb! ’<cd>)

- Signature ... (args ’event)
- Formula ... (pf ’event)
- Properties ... (props ’event)

- Print connected book directory (cbd)

- Rules whose top function symbol is ... (pl ’event)
- Rules stored by event ... (pr ’event)
- Rules stored by command ... (pr! ’<cd>)

- Monitored-runes (monitored-runes)

Control

- Load ... (ld filename)
- Accumulated Persistence
  - Activate (accumulated-persistence t)
  - Deactivate (accumulated-persistence nil)
  - Display statistics ordered by
    - frames (show-accumulated-persistence :frames)
    - times tried (show-accumulated-persistence :tries)
    - ratio (show-accumulated-persistence :ratio)
- Break rewrite
  - Start general rule monitoring (brr t)
  - Stop general rule monitoring (brr nil)
  - Print monitored runes (monitored-runes)
  - Monitor rune: ... (monitor ’(:definition <event>) ’t)
  - Unmonitor rune: ... (unmonitor ’(:definition <event>))
- Commands

Abort to ACL2 top-level #.
Term being rewritten :target
Substitution making :lhs equal :target :unify-subst
Hypotheses :hyps
Ith hypothesis ... :hyp <integer>
Left-hand side of conclusion :lhs
Right-hand side of conclusion :rhs
Type assumptions governing :target :type-alist
Ttree before :eval :initial-ttree
Negations of backchaining hyps pursued :ancestors

Rewrite's path from top clause to :target :path
Top-most frame in :path :top
Ith frame in :path ... :frame <integer>

AFTER :EVAL

Did application succeed? :wonp
Rewritten :rhs :rewritten-rhs
Ttree :final-ttree
Reason rule failed :failure-reason

CONTROL

Exit break :ok
Exit break, printing result :go
Try rule and re-enter break afterwards :eval

WITH NO RECURSIVE BREAKS

:ok! (:ok!)
:go! (:go!)
:eval! (:eval!)

WITH RUNES MONITORED DURING RECURSION

:ok ... (:ok$ sexpr)
:go ... (:go$ sexpr)
:eval ... (:eval$ sexpr)

Help (:help)

Enter ACL2 Loop (lp)
Quit to Common Lisp :Q
ABORT (:good-bye)

Settings

Mode

Logic (logic)
Program (program)
Guard checking on (set-guard-checking t)
Guard checking off (set-guard-checking nil)

Forcing

On (enable-forcing)
Off (disable-forcing)

Compile functions

On	(set-compile-fns t)
Off	(set-compile-fns nil)

Proof tree

Start prooftree	(start-proof-tree)pre (start-proof-tree nil))
Stop prooftree	(stop-proof-tree)post (stop-proof-tree))
Checkpoint forced goals on	(checkpoint-forced-goals)

Inhibit Display of

Error messages	(assign inhibit-output-lst ’(error))
Warnings	(assign inhibit-output-lst ’(warning))
Observations	(assign inhibit-output-lst ’(observation))
Proof commentary	(assign inhibit-output-lst ’(prove))
Proof tree	(assign inhibit-output-lst ’(prove))
Non-proof commentary	(assign inhibit-output-lst ’(event))
Summary	(assign inhibit-output-lst ’(summary))

Unused Variables

Ignore	(set-ignore-ok t)
Fail	(set-ignore-ok nil)
Warn	(set-ignore-ok :warn)

Irrelevant formulas

Ok	(set-irrelevant-formals-ok t)
Fail	(set-irrelevant-formals-ok nil)
Warn	(set-irrelevant-formals-ok :warn)

Load

Error action

Continue	(set-ld-error-actions :continue)
Return	(set-ld-error-actions :return)
Error	(set-ld-error-actions :error)

Error triples

On	(set-ld-error-triples t)
Off	(set-ld-error-triples nil)

Post eval print

On	(set-ld-post-eval-print t)
Off	(set-ld-post-eval-print nil)
Command conventions	(set-ld-post-eval-print :command-conventions)

Pre eval filter

All	(set-ld-pre-eval-filter :all)
Query	(set-ld-pre-eval-filter :query)

Prompt

On	(set-ld-prompt t)
Off	(set-ld-prompt nil)

Skip proofs

On	(set-ld-skip-proofs t)

Off	(set-ld-skip-proofs nil)

Verbose: on

On	(set-ld-verbose t)
Off	(set-ld-verbose nil)

Redefinition permitted	(redef)
Reset specials	(reset-ld-specials t)

HACKERS. DANGER!

RED redefinition!	(redef!)

Books

Print connected book directory	(cbd)
Set connected book directory ...	(set-cbd filename)
Certify-book ...	(certify-book filename)
Include-book ...	(include-book filename)

ACL2 Help

Documentation	(doc ’<symbol>)
Arguments	(args ’<symbol>)
More	(more)
Apropos ...	(docs ’<symbol>)
Info	(cl2-info)
Tutorial	(acl2-info-tutorial)
Release Notes	(cl2-info-release-notes)

3.4-B INFERIOR ACL2 POPUP MENU

Recent events	(pbt ’(:here -10))
Print Event	(pe ’<symbol>)
Print back to	(pbt ’<cd>)
Disable	(in-theory (disable <symbol>))
Enable	(in-theory (enable <symbol>))
Undo	(ubt ’:here)
Undo thru	(ubt ’<cd>)
Documentation	(doc ’<symbol>)
Arguments, etc	(args ’<symbol>)
Verify	Take clicked on form into interactive prover.

3.4-C KEYS

C-x C-e	Eval last sexp
C-c C-l	Load file
C-c C-a	Show arglist
C-c C-d	Describe symbol
C-c C-f	Show function documentation
C-c C-v	Show variable documentation
C-cl	Load file
C-ck	Compile file
C-ca	Show arglist
C-cd	Describe symbol
C-cf	Show function documentation

C-cv	Show variable documentation

The ACL2 Prooftree and Mouse Interface

Table of Contents