

# Theorem Proving with ACL2s



**Peter Dillinger**  
**Pete Manolios**



**Daron Vroon**



**J Strother Moore**

# ACL2

**A**  
**C**omputational  
**L**ogic  
for  
**A**pplicative  
**C**ommon  
**L**isp



**Bob Boyer**

**Matt Kaufmann**

**J Strother Moore**



THE UNIVERSITY OF TEXAS AT AUSTIN

# ACL2 Facts

- Latest (1989) theorem prover in the Boyer-Moore family (1971)
- “Industrial-strength” for the “applied logician”
- 2005 ACM Software System Award
- Semitriennial workshops
- Active user community

**AMD** 

*Rockwell  
Collins*





# ACL2 is ...

- **a programming language**
  - based on Common LISP; compilable & executable
  - total, first-order, side-effect free
- **a mathematical logic**
  - first-order; based on recursive functions
- **a mechanical theorem prover**
  - primarily based on term rewriting
  - written in ACL2; runs on Common LISP

# Using ACL2

```
peterd@cloud: ~
Use (help) to get some basic information on how to use GCL.
Temporary directory for compiler files set to /tmp/

ACL2 Version 3.3 built November 20, 2007 18:49:54.
Copyright (C) 2007 University of Texas at Austin
ACL2 comes with ABSOLUTELY NO WARRANTY. This is free software and you
are welcome to redistribute it under certain conditions. For details,
see the GNU General Public License.

Initialized with (INITIALIZE-ACL2 'INCLUDE-BOOK *ACL2-PASS-2-FILES*).
See the documentation topic note-3-3 for recent changes.
Note: We have modified the prompt in some underlying Lisps to further
distinguish it from the ACL2 prompt.

NOTE!! Proof trees are disabled in ACL2. To enable them in emacs,
look under the ACL2 source directory in interface/emacs/README.doc;
and, to turn on proof trees, execute :START-PROOF-TREE in the ACL2
command loop. Look in the ACL2 documentation under PROOF-TREE.

ACL2 Version 3.3. Level 1. Cbd "/shared/home/peterd/".
Distributed books directory "/nobak/peterd/acl2-3.3/books/".
Type :help for help.
Type (good-bye) to quit completely out of ACL2.

ACL2 !>
```

# Using ACL2



```
peterd@cloud: ~
Use (help) to get some basic information on how to use GCL.
Temporary directory for compiler files set to /tmp/

ACL2 Version 3.3 built November 20, 2007 18:49:54.
Copyright (C) 2007 University of Texas at Austin
ACL2 comes with ABSOLUTELY NO WARRANTY. This is free software and you
are welcome to redistribute it under certain conditions. For details,
see the GNU General Public License.

Initialized with (INITIALIZE-ACL2 'INCLUDE-BOOK *ACL2-PASS-2-FILES*).
See the documentation topic note-3-3 for recent changes.
Note: We have modified the prompt in some underlying Lisps to further
distinguish it from the ACL2 prompt.

NOTE!! Proof trees are disabled in ACL2. To enable them in emacs,
look under the ACL2 source directory in interface/emacs/README.doc;
and, to turn on proof trees, execute :START-PROOF-TREE in the ACL2
command loop. Look in the ACL2 documentation under PROOF-TREE.

ACL2 Version 3.3. Level 1. Cbd "/shared/home/peterd/".
Distributed books directory "/nobak/peterd/acl2-3.3/books/".
Type :help for help.
Type (good-bye) to quit completely out of ACL2.

ACL2 !>
```



**+** **Years**



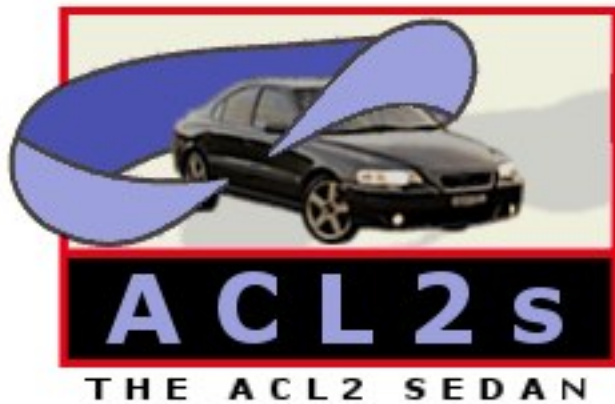
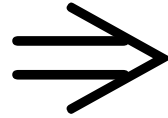
# The Emacs Sentiment

- **Loved by experts**
  - extremely customizable
  - mouse/GUI not required
- **Difficult for novices**
  - unfamiliar key bindings
  - efficient use comes slowly
  - user must maintain link between source code and theorem prover state

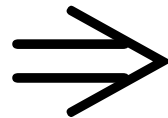
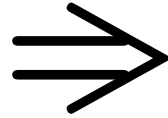
# “The ACL2 Sedan”



# “The ACL2 Sedan”



# “The ACL2 Sedan”



# “The ACL2 Sedan”



The screenshot shows the Eclipse IDE interface for ACL2 development. The main editor displays the source code for `*lights.lisp` in ACL2s Mode. The code includes several function definitions for manipulating configurations and proving theorems. The command window shows the execution of a goal, resulting in a subgoal that has been simplified. The proof tree view on the right shows the current state of the proof, including the goal and the rules used to simplify it.

```
(t
  (cons (car config)
        (flip (1- row) col (cdr config))))))

(defcong config-row-equal config-row-equal (flip-col c row)
  (defcong config-equal config-equal (flip row col config) 3)

(defthm flip-cols-commute
  (config-row-equal (flip-col c1 (flip-col c2 row))
                    (flip-col c2 (flip-col c1 row))))

(defthm flips-commute
  (config-equal (flip r1 c1 (flip r2 c2 config))
                (flip r2 c2 (flip r1 c1 config))))

(defthm double-flip-col
  (config-row-equal (flip-col col (flip-col col config))
                    config))

(defthm double-flip
  (config-equal (flip row col (flip row col config))
                config))

(defun move (row col config)
  (let* ((config (flip row col config))
         (config (flip (1+ row) col config))
         (config (flip (1- row) col config))
         (config (flip row (1+ col) config))
         (config (flip row (1- col) config)))
```

Ready for command input  
the following goal.

Subgoal \*1/2'4'  
(IMPLIES (CONSP (CONS CONFIG1 CONFIG2))  
 CONFIG1).

This simplifies, using primitive type r

Subgoal \*1/2'5'  
NIL.  
^^^ Checkpoint Subgoal \*1/2'5' ^^^

Summary  
Form: ( DEFTHM DOUBLE-FLIP-COL ... )  
Rules: ( (:COMPOUND-RECOGNIZER NATP-COMP  
 (:DEFINITION =)  
 (:DEFINITION CONFIG-ROW-EQUAL)  
 (:DEFINITION ENDP)  
 (:DEFINITION FLIP-COL)  
 (:DEFINITION IFF)  
 (:DEFINITION NOT)  
 (:DEFINITION SYNPN)  
 (:ELIM CAR-CDR-ELIM)  
 (:EQUIVALENCE CONFIG-ROW-EQUAL-  
 (:EXECUTABLE-COUNTERPART BINARY  
 (:EXECUTABLE-COUNTERPART EQUAL)  
 (:EXECUTABLE-COUNTERPART NATP)  
 (:EXECUTABLE-COUNTERPART NOT)  
 (:FAKE-RUNE-FOR-TYPE-SET NIL)  
 (:INDUCTION FLIP-COL)  
 (:REWRITE CAR-CONS)

Tracking lights.lisp.a2s  
( DEFTHM DOUBLE-FLIP-COL ... )  
\*\*\*\*\* FAILED \*\*\*\*\*  
c0 Goal PUSH \*1  
\*\*\*\*\*  
c3 \*1 INDUCT  
1 |Subgoal \*1/2' preproc  
1 ||Subgoal \*1/2' simp  
1 |||Subgoal \*1/2'' sir  
1 ||||Subgoal \*1/2''' f  
1 |||||Subgoal \*1/2'4'  
c0 |||||Subgoal \*1/2'5'

# Outline

- Introduction
  - **Basic ACL2/ACL2s demo**
  - **Supporting interaction + demo**
  - **Other features + demo**
  - **Effectiveness, Related/Future Work**
- 
- **Efficient execution in ACL2 + demo**

# **ACL2/ACL2s “Basic” Demo**

# Outline

- Introduction
  - Basic ACL2/ACL2s demo
  - **Supporting interaction + demo**
  - **Other features + demo**
  - **Effectiveness, Related/Future Work**
- 
- **Efficient execution in ACL2 + demo**

# UI Requirements for ACL2s

- **“Script management” interaction**  
(undo mechanism)
- **Command line interaction**  
(categorization)
- **Mixed SM/CL interaction**  
(fix consistency)
- **Better input/output abstractions**  
(awareness+checking+markup)

# **ACL2s “Interaction” Demo**

# Outline

- Introduction
  - Basic ACL2/ACL2s demo
  - Supporting interaction + demo
  - **Other features + demo**
  - **Effectiveness, Related/Future Work**
- 
- **Efficient execution in ACL2 + demo**

# Other Features

- **Session modes**
- **Enhanced termination analysis (CCG)**
- **Editor features (indent, match, color)**
- **“Proof tree” view**
- **One-click certification**
- **Basic graphics programming**

# **ACL2s “Other Features” Demo**

# Outline

- Introduction
  - Basic ACL2/ACL2s demo
  - Supporting interaction + demo
  - Other features + demo
  - **Effectiveness, Related/Future Work**
- 
- **Efficient execution in ACL2 + demo**

# ACL2s Effectiveness

- **Used in several classes**
  - **UT (Prof. Moore) and GT/NE (Prof. Manolios)**
  - **Total of ~150 students**
- **Sense:**
  - **Lower start-up cost (install, configure, get acquainted) vs. Emacs**
  - **Easily understood with a brief demo**
- **A few comments, a few bugs (fixed)**

# Related Work

- **Proof General**
  - Basic hook for ACL2
  - PG/Eclipse
- **DrACuLa (ACL2 in Dr. Scheme)**
  - ✗ Incomplete (focus *only* on novices)
  - ✓ Debugging support
    - steppers, visualizations, etc.



# Future Work

- **Debugging support (eval and proofs)**
- **More GUI, less text**
- **Further improve termination analysis**

# Outline

- Introduction
  - Basic ACL2/ACL2s demo
  - Supporting interaction + demo
  - Other features + demo
  - Effectiveness, Related/Future Work
- 
- **Efficient execution in ACL2 + demo**

# **ACL2 “Efficient execution” Demo**

# Thanks

**ACL2s (free):**  
**<http://acl2s.peterd.org>**  
**(or ask Google)**

**ACL2 (free):**  
**<http://www.cs.utexas.edu/users/moore/acl2/>**  
**(or ask Google)**