ACL2 Support for Automated and Interactive Proof

Matt Kaufmann
The University of Texas at Austin
Dept. of Computer Science

KeY Workshop, July 27, 2015

OUTLINE

OUTLINE

INTRODUCTION (1)

Quoting the ACL2 home page:

> *ACL2 is a logic and programming language in which you can model computer systems, together with a tool to help you prove properties of those models. "ACL2" denotes "***A** **C***omputational* **L***ogic for* **A***pplicative* **C***ommon* **L***isp".*

Goal for this talk:
Give a sense of the ACL2 system, especially how it supports a combination of **automated proof** and **user interaction**.

BUT HOW CAN I ACHIEVE THIS GOAL?

My answer ....

INTRODUCTION (2)

I DON'T KNOW!

But here is my plan.

- **BACKGROUND**: Present basics, history, usage, etc.
- **DEMOS**: Give a sense of how ACL2 can be used.
- **REWRITING**: Summarize the primary way to control the ACL2 prover.
- **OTHER STUFF**: Survey *briefly* other ways that ACL2 supports user interaction.

And the most important thing:

- **Please ask questions during the talk!**

Outline

OUTLINE

Background: Overview

- ACL2 is freely available, including libraries of *certifiable books*
- Let's explore the ACL2 home page.
- ACL2 is written mostly in itself (!).
  - About 10 MB of source code (Version 7.1).
- *Bleeding edge* for libraries (*community books*) and the ACL2 system are available from Github.
  - Well over 400,000 *events* (theorems, definitions, other) are evaluated in the community books.
- Workshop series: #13 is at UT, Oct. 1-2, 2015.

Background: Overview (continued)

Development history:

- Bob Boyer and J Moore started ACL2 in 1989. I joined and Bob dropped out in 1993. J and I continue its development.
- *Boyer-Moore Theorem Provers* go back to the start of their collaboration in 1971.

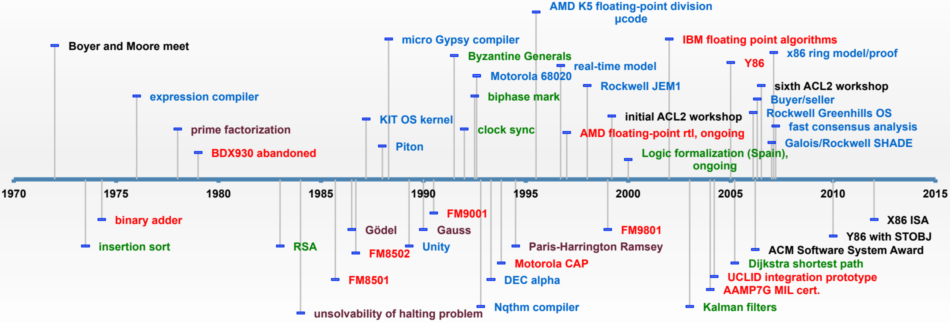
Industrial usage: As far as I know, ACL2 is the only interactive theorem prover (ITP) used with some regularity at several companies:

- AMD, Centaur, IBM, Intel, Oracle, Rockwell Collins

There are also users in the **U.S. Government** and **universities**, including —

- UT Austin: x86 interpreter defined in ACL2, validation by co-simulation, proofs about x86 machine code

PARTIAL TIMELINE



BACKGROUND: LOGIC

The ACL2 logic is a first-order logic with induction up to ε_0.

But all ACL2 theories extend a given *ground-zero* theory, which axiomizes data types for:

- numbers (complex rationals), characters, strings, symbols;
- trees and lists, using a pairing operation (`cons`).

ACL2 extensions are *conservative* (a demo will discuss this).

- See M. Kaufmann and J Moore, "Structured Theory Development for a Mechanized Logic." *Journal of Automated Reasoning* 26, no. 2 (2001) 161-203.

BACKGROUND: STRENGTHS

- Proof automation
- Support for user interaction
- Fast execution
- Documentation (about 100,000 lines for just the system)
- Interfaces include **Emacs**
  (Is that really an interface? A strength?)
  and the Eclipse-based ACL2 Sedan.

A potential weakness: first-order logic with only basic quantifier support (but recursion helps).

BACKGROUND: MORE INFORMATION

NOTE: A longer variant of this talk, but targeted to CS grad students and with more focus on *using* ACL2, is here:

http://www.cs.utexas.edu/users/kaufmann/talks/acl2-intro-2015-04/acl2-intro.pdf

That talk mentions this link to several demos and their logs:

http://www.cs.utexas.edu/users/kaufmann/talks/acl2-intro-2015-04/demos.tgz

Outline

OUTLINE

DEMOS (I)

- All demos today, with logs, are in the gzipped tar file `demos.tgz` in this directory.

- ACL2 programming and evaluation
  [DEMO]: file `demo-1.lsp`
  (log `demo-1-log.txt`)

- ACL2 as an automatic theorem prover
  [DEMO]: file `demo-2.lsp`
  (log `demo-2-log.txt`)
  - ACL2 provides automation for induction, linear arithmetic, Boolean reasoning, rule application, . . .
  - . . . but lemmas are usually needed (sometimes from libraries).

DEMOS (I) (CONTINUED)

ACL2 supports formally verified extensions.

In particular, GL is a *verified clause processor* defined and verified by an ACL2 user, Sol Swords.
GL does proofs about finite domains by *bit-blasting*.

The next demo illustrates GL. It also shows the use of LOCAL, for "private" events (using *conservativity*).

[DEMO]: book `demo-gl.lisp`
(log `demo-gl-log.txt`)

Outline

OUTLINE

Rewriting in ACL2

ACL2 is typically controlled with conditional rewrite rules, although there are other rule-classes too.

The basic idea: the ACL2 rewriter automatically applies the rule

$$H \longrightarrow L = R$$

by replacing an instance L/s of L by R/s, when the rewriter can verify H/s.

The documentation topic for rewrite shows many ways to control the rewriter (needed only occasionally). I'll mention only a few:

REWRITING IN ACL2 (2)

- backchain-limit: limit effort to relieve hypotheses
- force: defer proving H if necessary
- hide: hide a term from the rewriter
- syntaxp: attach a heuristic filter on a rule

Example of `syntaxp`: consider $3 + (4 + x)$.
It's already in normal form: right-associated.
Our wish: $3 + (4 + x) = (3 + 4) + x = 7 + x$.

```
ACL2 !>:pe associativity-of-+
        -997  (DEFAXIOM ASSOCIATIVITY-OF-+
                        (EQUAL (+ (+ X Y) Z) (+ X (+ Y Z))))

ACL2 !>:pe fold-consts-in-+
        -158  (DEFTHM FOLD-CONSTS-IN-+
                      (IMPLIES (AND (SYNTAXP (QUOTEP X))
                                    (SYNTAXP (QUOTEP Y)))
                               (EQUAL (+ X (+ Y Z)) (+ (+ X Y) Z))))
ACL2 !>
```

REWRITING IN ACL2 (3)

$$H \longrightarrow L = R \qquad \textit{(ordinary rewrite rule)}$$

$$H \longrightarrow L \sim R \qquad \textit{(congruence-based rewrite rule)}$$

where $\sim$ is an equivalence relation.

Users prove equivalence, congruence, and refinement rules, to tell ACL2 when such rewrites are valid.

B. Brock, M. Kaufmann, and J S. Moore. Rewriting with Equivalence Relations in ACL2. *J. Aut. Reasoning* 40 (2008).

M. Kaufmann and J S. Moore. Double Rewriting for Equivalential Reasoning in ACL2. *Proc. ACL2 Workshop 2006*.

M. Kaufmann and J S. Moore. Rough Diamond: An Extension of Equivalence-based Rewriting. *Proc. ITP 2014*.

Outline

Outline

DEMOS (II)

Time permitting . . .

These demos are excerpted from my TPHOLs 2008 talk.

NOTE: Here I'll just show highlights, to give an impression about ACL2 usage.

- JVM demo
- Our final demo illustrates "the method" recommended for dealing with proof failures.

OUTLINE

Outline

Very Brief Survey of ACL2 Features

Let's see a bit more about how ACL2 supports proof development . . .

- . . . by exploring briefly the ACL2 documentation.

NOTE:
I would be very happy to elaborate on any of these topics!

Very Brief Survey of ACL2 Features (2)

In particular, we might explore a few debugging features, as time and interest permit.

- accumulated-persistence
- break-rewrite
- cgen
- cw-gstack
- disassemble$
- dmr
- failed-forcing
- failure
- find-lemmas
- forward-chaining-reports
- guard-debug
- measure-debug
- nil-goal
- print-gv
- profile-acl2
- profile-all
- proof-checker
- proof-tree
- pstack
- quick-and-dirty-subsumption-replacement-step
- redo-flat
- remove-hyps
- set-debugger-enable
- set-guard-msg
- sneaky
- spacewalk
- splitter
- time-tracker
- trace
- walkabout
- watch

OUTLINE

Outline

CONCLUSION

- ACL2 has a 25 (or 44) year history and is used in industry.

  > *"Microprocessor design goes daily through numerous optimizations that affect thousands of lines of code. These optimizations must be proved correct."*

  — Anna Slobodova, verification manager at Centaur Technology
- ACL2 provides automation but scales to large problems …
  … with libraries and with user guidance.
- For more information, see the ACL2 home page.

THANK YOU!