Automated Reasoning and The ACL2 Theorem Proving System

J Strother Moore
Matt Kaufmann
Department of Computer Sciences
University of Texas at Austin
{moore,kaufmann}@cs.utexas.edu

The 2006 Visions of Computing Lecture Series
Department of Computer Sciences
University of Texas at Austin
November 9, 2006

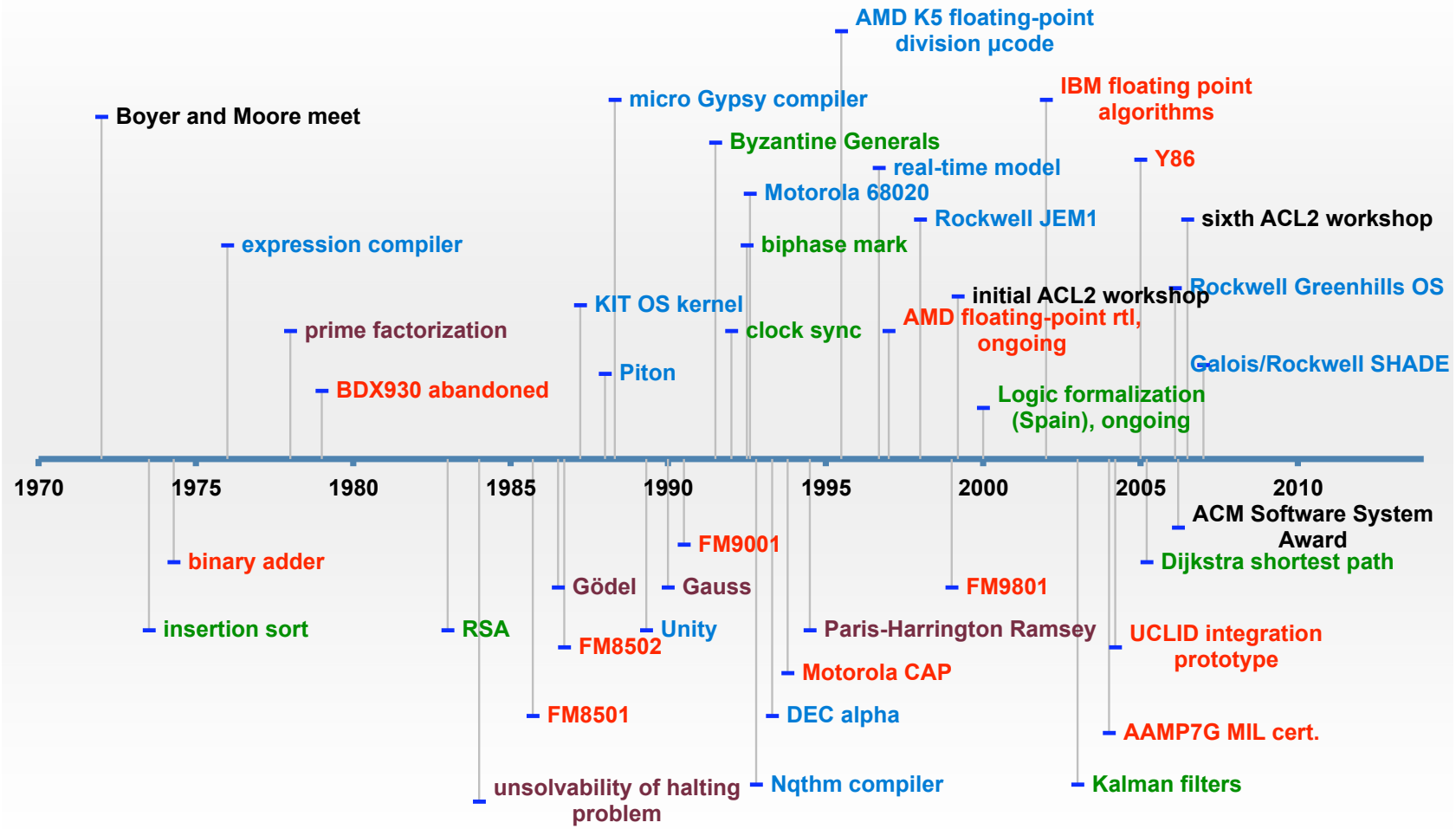
Machines that Reason: The Big Picture

Instead of debugging a program, one should prove that it meets its specifications, and this proof should be checked by a computer program.

— John McCarthy, *“A Basis for a Mathematical Theory of Computation,” 1961*

In order for a program to be capable of learning something it must first be capable of being told it.

— John McCarthy, *“Programs with Common Sense” otherwise known as “The Advice Taker paper”, 1959.*


AMD K5 floating-point division µcode
IBM floating point algorithms
micro Gypsy compiler
Boyer and Moore meet
Byzantine Generals
Y86
real-time model
Motorola 68020
Rockwell JEM1
sixth ACL2 workshop
expression compiler
biphase mark
Rockwell Greenhills OS
initial ACL2 workshop
KIT OS kernel
AMD floating-point rtl, ongoing
prime factorization
clock sync
Galois/Rockwell SHADE
Piton
BDX930 abandoned
Logic formalization (Spain), ongoing
1970
1975
1980
1985
1990
1995
2000
2005
2010
ACM Software System Award
FM9001
binary adder
Dijkstra shortest path
Gödel
Gauss
FM9801
insertion sort
RSA
Unity
Paris-Harrington Ramsey
UCLID integration prototype
FM8502
Motorola CAP
FM8501
DEC alpha
AAMP7G MIL cert.
unsolvability of halting problem
Nqthm compiler
Kalman filters

Key Research Problems

1. Automatic Invention of Lemmas and New Concepts
2. How to use Examples and Counterexamples
3. How to use Analogy, Learning, and Data Mining
4. How to Architect an Open Verification Environment
5. Parallel, Distributed and Collaborative Theorem Proving
6. User Interface and Interactive Steering
7. Education of the User Community – and Their Managers
8. How to Build a Verified Theorem Prover

If we had some exact language . . . or at least a kind of truly philosophic writing, in which the ideas were reduced to a kind of alphabet of human thought, then all that follows rationally from what is given could be found by a kind of calculus, just as arithmetical or geometrical problems are solved.

— Leibniz (1646–1716)

The ACL2 Theorem Prover

Introduction

ACL2 = ACL^2 = ACLACL

A Computational Logic for Applicative Common Lisp

Following in the tradition of past Boyer-Moore provers, ACL2 is:

- a first-order logic with mathematical induction;
- a functional programming language; and
- a computer program based on these, which can prove and organize theorems.

For more information, see the ACL2 home page, `http://www.cs.utexas.edu/users/moore/acl2/`:

- Downloads
- Tutorials and documentation
- Weekly UT seminar
- Proceedings of the six past workshops
- Papers
- Mailing lists

Introduction

ACL2 = ACL^2 = ACLACL

A Computational Logic for Applicative Common Lisp

Following in the tradition of past Boyer-Moore provers, ACL2 is:

- a first-order logic with mathematical induction;
- a functional programming language; and
- a computer program based on these, which can prove and organize theorems.

For more information, see the ACL2 home page, `http://www.cs.utexas.edu/users/moore/acl2/`:

- Downloads
- Tutorials and documentation
- Weekly UT seminar
- Proceedings of the six past workshops
- Papers
- Mailing lists

Introduction

ACL2 = ACL^2 = ACLACL

A Computational Logic for Applicative Common Lisp

Following in the tradition of past Boyer-Moore provers, ACL2 is:

- a first-order logic with mathematical induction;
- a functional programming language; and
- a computer program based on these, which can prove and organize theorems.

For more information, see the ACL2 home page, `http://www.cs.utexas.edu/users/moore/acl2/`:

- Downloads
- Tutorials and documentation
- Weekly UT seminar
- Proceedings of the six past workshops
- Papers
- Mailing lists

System Development

- Begun in August 1989 by Bob Boyer and J Moore, continuing their line of research in automated reasoning.
- Further research and development by Matt Kaufmann and J Moore since 1993, with contributions from many others (though we are responsible for bugs!)
- 8.4M of source files, programmed primarily in itself: forces attention to sufficient functional language features and efficient execution
- Has evolved with user feedback
- Distribution contains about 40,000 theorems, many from libraries of user-contributed *books* (input files)
  - Serves as a test suite for the ACL2 system

System Development

- Begun in August 1989 by Bob Boyer and J Moore, continuing their line of research in automated reasoning.
- Further research and development by Matt Kaufmann and J Moore since 1993, with contributions from many others (though we are responsible for bugs!)
- 8.4M of source files, programmed primarily in itself: forces attention to sufficient functional language features and efficient execution
- Has evolved with user feedback
- Distribution contains about 40,000 theorems, many from libraries of user-contributed *books* (input files)
  - Serves as a test suite for the ACL2 system

System Development

- Begun in August 1989 by Bob Boyer and J Moore, continuing their line of research in automated reasoning.
- Further research and development by Matt Kaufmann and J Moore since 1993, with contributions from many others (though we are responsible for bugs!)
- 8.4M of source files, programmed primarily in itself: forces attention to sufficient functional language features and efficient execution
- Has evolved with user feedback
- Distribution contains about 40,000 theorems, many from libraries of user-contributed *books* (input files)
  - Serves as a test suite for the ACL2 system

System Development

- Begun in August 1989 by Bob Boyer and J Moore, continuing their line of research in automated reasoning.
- Further research and development by Matt Kaufmann and J Moore since 1993, with contributions from many others (though we are responsible for bugs!)
- 8.4M of source files, programmed primarily in itself: forces attention to sufficient functional language features and efficient execution
- Has evolved with user feedback
- Distribution contains about 40,000 theorems, many from libraries of user-contributed *books* (input files)
  - Serves as a test suite for the ACL2 system

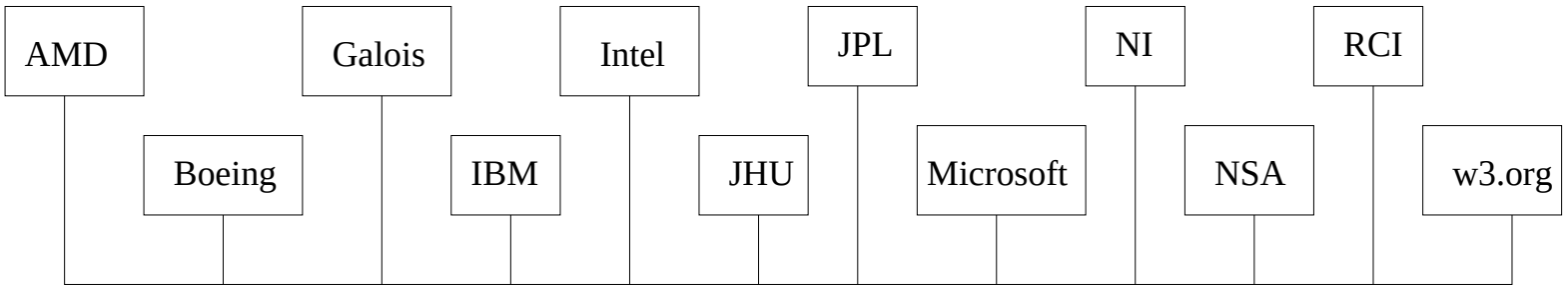
System Development

- Begun in August 1989 by Bob Boyer and J Moore, continuing their line of research in automated reasoning.
- Further research and development by Matt Kaufmann and J Moore since 1993, with contributions from many others (though we are responsible for bugs!)
- 8.4M of source files, programmed primarily in itself: forces attention to sufficient functional language features and efficient execution
- Has evolved with user feedback
- Distribution contains about 40,000 theorems, many from libraries of user-contributed *books* (input files)
  - Serves as a test suite for the ACL2 system

System Development

- Begun in August 1989 by Bob Boyer and J Moore, continuing their line of research in automated reasoning.
- Further research and development by Matt Kaufmann and J Moore since 1993, with contributions from many others (though we are responsible for bugs!)
- 8.4M of source files, programmed primarily in itself: forces attention to sufficient functional language features and efficient execution
- Has evolved with user feedback
- Distribution contains about 40,000 theorems, many from libraries of user-contributed *books* (input files)
  - Serves as a test suite for the ACL2 system

The ACL2 Community



"Customers"

Our Research Program

ACL2 PROJECT

ACL2 System

Application−Oriented Research

Logical Issues

ACL2 users are, in essence, applied logicians.

But we ACL2 system builders are also applied logicians!

- ACL2 provides proof structuring constructs that have rigorous logical foundations. These constructs have previously been a source of subtle logical bugs.
- We are developing sound connections with other proof tools.
- ACL2 processes some of its own source code (falls short of formally verifying ACL2, but it’s a start...).

User Support

Our goal: Incorporate our research into industrial-strength software.

- Provide hypertext documentation (over 1200 pages if printed).
- Support the help list.
- Generate interactive feedback, in particular:
  - robust error and warning messages; and
  - proof logs with a navigation tool.
- Respond to user requests, which drive many of the improvements.

Efficiency Issues

Example: Recent changes in theory management (collections of rules) to support users at Rockwell Collins, Inc.

Some statistics from one of their proofs:

ACL2 3.0:	3156.21 secs then FAILS (out of memory)
ACL2 3.0.1:	3095.68 secs
ACL2 3.0.2 alpha:	265.00 secs

Some Current Directions

- Connect with other reasoning tools (e.g., SAT and SMT solvers and resolution provers) to leverage progress in the field.
- Do more search during proof process, while maintaining controllability.
- Incorporate memoization and hash-table work pursued by UT CS professors Bob Boyer and Warren Hunt.
- Support development of interfaces for use in undergraduate courses.
- And as always, support our users.

Some Current Directions

- Connect with other reasoning tools (e.g., SAT and SMT solvers and resolution provers) to leverage progress in the field.
- Do more search during proof process, while maintaining controllability.
- Incorporate memoization and hash-table work pursued by UT CS professors Bob Boyer and Warren Hunt.
- Support development of interfaces for use in undergraduate courses.
- And as always, support our users.

Some Current Directions

- Connect with other reasoning tools (e.g., SAT and SMT solvers and resolution provers) to leverage progress in the field.
- Do more search during proof process, while maintaining controllability.
- Incorporate memoization and hash-table work pursued by UT CS professors Bob Boyer and Warren Hunt.
- Support development of interfaces for use in undergraduate courses.
- And as always, support our users.

Some Current Directions

- Connect with other reasoning tools (e.g., SAT and SMT solvers and resolution provers) to leverage progress in the field.
- Do more search during proof process, while maintaining controllability.
- Incorporate memoization and hash-table work pursued by UT CS professors Bob Boyer and Warren Hunt.
- Support development of interfaces for use in undergraduate courses.
- And as always, support our users.

Some Current Directions

- Connect with other reasoning tools (e.g., SAT and SMT solvers and resolution provers) to leverage progress in the field.
- Do more search during proof process, while maintaining controllability.
- Incorporate memoization and hash-table work pursued by UT CS professors Bob Boyer and Warren Hunt.
- Support development of interfaces for use in undergraduate courses.
- And as always, support our users.

Some Current Directions

- Connect with other reasoning tools (e.g., SAT and SMT solvers and resolution provers) to leverage progress in the field.
- Do more search during proof process, while maintaining controllability.
- Incorporate memoization and hash-table work pursued by UT CS professors Bob Boyer and Warren Hunt.
- Support development of interfaces for use in undergraduate courses.
- And as always, support our users.

Conclusion

We are excited by our progress to date.

We thank Warren Hunt for repeatedly pursuing applications that have pushed the envelope.

More generally, we want to acknowledge the ACL2 user community, without whom progress would be much less.

We invite you to use ACL2!

`http://www.cs.utexas.edu/users/moore/acl2/`