Reasoning about Digital Artifacts with ACL2

J. Strother Moore

Department of Computer Science
1616 Guadalupe Street, Suite 2.408
University of Texas at Austin
Austin, TX 78701
moore@cs.utexas.edu

Abstract

ACL2 is both a programming language in which computing systems can be modeled and a tool to help a designer prove properties of such models. ACL2 stands for "*A* *C*omputational *L*ogic for *A*pplicative *C*ommon *L*isp" and provides mechanized reasoning support for a first-order axiomatization of an extended subset of functional Common Lisp. Most often, ACL2 is used to produce operational semantic models of artifacts. Such models can be executed as functional Lisp programs and so have dual use as both pre-fabrication simulation engines and as analyzable mathematical models of intended (or at least designed) behavior.

This project had its start 40 years ago in Edinburgh with the first Boyer-Moore Pure Lisp theorem prover and has evolved from proofs about list concatenation and reverse to proofs about industrial models.

Industrial use of theorem provers to answer design questions of critical importance is so surprising to people outside of the theorem proving community that it bears emphasis. In the 1980s, the earlier Boyer-Moore theorem prover, Nqthm, was used to verify the "Computational Logic stack" – a hardware/software stack starting with the NDL description of the netlist for a microprocessor and ascending through a machine code ISA, an assembler, linker, and loader, two compilers (for subsets of Pascal and Lisp), an operating system, and some simple applications. The system components were proved to compose so that properties proved of high-level software were guaranteed by the binary image produced by the composition. At around the same time, Nqthm was used to verify 21 of the 22 subroutines in the MC68020 binary machine code produced from the Berkeley C String Library by `gcc -o`, identifying bugs in the library as a result.

Applications like these convinced us that (a) industrial scale formal methods was practical and (b) Nqthm's Pure Lisp produced uncompetitive results compared to C when used for simulation engines. We therefore designed ACL2, which initially was Nqthm recoded to support applicative Common Lisp.

The 1990s saw the first industrial application of ACL2, to verify the correspondence between a gate-level description of the Motorola CAP DSP and its microcode engine. The Lisp model of the microcode engine was proved to be bit- and cycle-accurate but operated several times faster than the gate-level simulator in C because of the competitive execution speed of Lisp and the higher level of trusted abstraction. Furthermore, it was used to discover previously unknown microcode hazards. An executable Lisp predicate was verified to detect all hazards and subsequently used by microcode programmers to check code. This project and a subsequent one at AMD to verify the floating point division operation on the AMD K5 microprocessor demonstrated the practicality of ACL2 but also highlighted the need to develop better Lisp system programming tools wedded to formal methods, formal modeling, proof development, and "proof maintenance" in the face of evolution of the modeled artifacts.

Much ACL2 development in first decade of the 21st century was therefore dedicated to such tools and we have witnessed a corresponding increase in the use of ACL2 to construct and reason about commercial artifacts. ACL2 has been involved in the design of all AMD desktop microprocessors since the Athlon; specifically, ACL2 is used to verify floating-point operations on those microprocessors. Centaur Technology (chipmaker for VIA Technologies) uses ACL2 extensively in verifying its media unit and other parts of its x86 designs. Researchers at Rockwell-Collins have shown that ACL2 models of microprocessors can run at 90% of the speed of C models of those microprocessors. Rockwell-Collins has also used ACL2 to do information flow proofs to establish process separation for the AAMP7G cryptoprocessor and, on the basis of those proofs, obtained MILS certification using Formal Methods techniques as specified by EAL-7 of the Common Criteria. IBM has used ACL2 to verify floating point operations on the Power 4 and other chips. ACL2 was also used to verify key properties of the Sun Java Virtual Machine's class loader.

In this talk I will sketch the 40 year history of this project, showing how the techniques and applications have grown over the years. I will demonstrate ACL2 on both some simple problems and a complicated one, and I will deal briefly with the question of how – and with what tool – one verifies a verifier. For scholarly details of some of how to use ACL2 and some of its industrial applications see [1, 2]. For source code, lemma libraries, and an online user's manual, see the ACL2 home page, http://www.cs.utexas.edu/users/moore/acl2.

Categories and Subject Descriptors F.4.1 [*Mathematical Logic*]: Mechanical theorem proving; F.3.2 [*Semantics of Programming Languages*]: Operational semantics, Program analysis

General Terms Algorithms, Design, Languages, Reliability, Security, Theory, Verification

Keywords automatic theorem proving, hardware verification, JVM, microprocessor verification, operational semantics, software stack, virtual machine verification

Acknowledgments

I thank the entire user community of ACL2 and the "Boyer-Moore theorem provers," and especially Bob Boyer, Matt Kaufmann, and Warren Hunt, without whose inspired work and dedication the results reported here would be but wishful thinking. I also thank the many industrial organizations that have put their trust in us, especially Advanced Micro Devices, Centaur, Freescale, IBM, Motorola, and Rockwell-Collins. Finally, I thank the sponsors of the University of Texas ACL2 group, in particular, the National Science Foundation under Grant Numbers CCF-0945316, CNS-0910913, and CCF-0916772, DARPA contract N66001-10-2-4087, SRC contract 08-TJ-1849, and ForrestHunt, Inc. Any opinions, findings, and conclusions or recomendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] M. Kaufmann, P. Manolios, and J. S. Moore, editors. *Computer-Aided Reasoning: ACL2 Case Studies*. Kluwer Academic Press, Boston, MA., 2000.

[2] M. Kaufmann, P. Manolios, and J. S. Moore. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Press, Boston, MA., 2000.