The Strengths of ACL2 and Its Path to the Future

Warren A. Hunt, Jr. and Anna Slobodova
UT Austin Arm Inc.
hunt@cs.utexas.edu anna.slobodova@arm.com

May, 2025

Outline

We want to initiate a dialog about the future of ACL2.

1. Opening discussion
2. Strengths of ACL2
3. Usage of ACL2
4. Summary and open questions

Opening discussion

- We have used ACL2 for decades - as we are supporters of the system
  - What do you hope to do with ACL2 in the future?
  - What do you believe would make ACL2 more attractive to potential users?
- When asked to talk about the present and future, we turned to community. This is a compilation of what we heard from you (either by E-mail or through an interview. Thanks to:
  - Alessandro Coglio
  - David Hardin
  - Grant Jurgenson
  - Andrei Koltsov
  - Pete Manolios
  - Mihir Mehta
  - Sudarshan Srinivasan
  - Eric Smith
- There is no successful system without an engaged user base
- Consider this forum as an opening of the discussion.
- Let us know what we missed, and what we have wrong.

Strengths of ACL2

- It has great support for **automated induction**
- It enables **high-performance, executable** models
- It has a **fast interactive prover** environment
- It's **extensible** through verified clause processors and APIs
- **Extensive** hyper-linked documentation
- **Training** and **education** material
- Great **support** by the ACL2 community

Usage of ACL2

- Language modeling (e.g., x86 ISA, Java, C, Rust, Algorithmic C)
- Proving code execution and transformations correct
- Modeling micro-architecture specification at Arm, Centaur Technology, and Intel
- Analyzing hardware designs (Verilog, SystemVerilog) and proving them correct with respect to micro-architectural specifications
- Modeling of rapid, single-flux, quantum circuit models.
- Teaching vehicle for logic and reasoning for undergraduate and graduate classes (ACL2 and ACL2s) – 28 new users trained this semester at UT, similar number at Northeastern University
- Modelling of algorithms and proving them correct (e.g. linear-algebra algorithms, math algorithms, crypto)
- Mechanizing existing mathematical theories and their proofs (ACL2 and ACL2r)

An example of multi-functional use of ACL2 at Centaur Technology/Intel

- Specification and verification for high-performance x86 designs
- ACL2 as scripting language
  - to extract information from project data-bases about instruction encoding, latency and where they will be executed (assignment to an execution unit)
  - to track progress of work (what proofs have been done, which are pending, what bugs where filed)
  - to generate debug scripts for designers (connecting to JasperGold)
  - to generate specifications of micro-operations and their proofs
- ACL2 as specification of micro-operations (several thousand)
- ACL2 as a model of SystemVerilog design (at Centuar Technology and Intel)
- ACL2 as verification tool through its verified clause processor FGL
- ACL2 as documentation tool - gathering micro-operation specifications, proofs and their assumptions on one website, making it aproachable by designers
- USing one langugage for specification and model, allows a seemless transition from hardware verification to micro-code verification

The ACL2 Ecosystem

ACL2 is a *Swiss Army Knife* for the specification, verification, and automated processing of industrial-sized computer models.

When used by industry, ACL2 become a part of a *flow* in the spirit of Python scripts to ingest content (from designers and other tools) and produce output for users and downstream tools.

As far as we know, ACL2 remains the most capable system for analyzing hardware designs specified in Verilog.

With the inclusion of floating-point arithmetic in ACL2, we believe ACL2 should be considered as a replacement for important FORTRAN codes, such as weather modeling.

Commercial industry would benefit greatly from having a general-purpose, executable, formal specification system – that also provided an analytical capability. ACL2 can assist with this vision.

But ACL2 faces headwinds such as text-based I/O facilities do not inspire recently-trained engineers, which limits the supply of skilled ACL2 users.

Summary and open questions

- ACL2 is a great tool with excellent scripting and analysis performance
- ACL2 has friendly community that helps newcomers
- There is always space to **improve** as we heard from all of you:
  - Develop a better (and more modern) editor interface
  - Improve the ACL2 web site
  - Improve promotion of ACL2, e.g. by adding conference presentations and papers (or at least references to them) for any work done using ACL2 to ACL2 website
  - Engage the younger generation (academia - teach classes in ACL2, industry - hire junior people and train them)
  - Clean-up/merge some books and avoid duplicate efforts (can we engage AI in this task?)
  - Expand the pool of maintainers that will assure continuation. Original developers are still around and willing to help.
  - Additional suggestions welcome...

Summary and open questions

- How can we improve the ACL2 business model? In the past most maintenance was supported by FHI and most funds came from Centaur Technology and later Intel.
- Other open-source tools like CVC5 maintain it using gifts from the companies that use it to Professor Clark Barrett's research lab. It would be beneficial if the users that use ACL2 as part of their job set aside such gifts.
- Other option would be to set aside resources as part of the overhead in the contracts.
- Do we want to revive regular ACL2 seminars (online for the whole community)?

You Can Be Engaged!

How can you help? It takes a village!

- by organizing an ACL2 Workshop or participaiting in the review process
- by presenting your work to ACL2 community in an online seminar
- by advertising ACL2
- by participating in improving ACL2 public libraries
- by getting some resources for ACL2 maintenance
- by just doing good work using ACL2