ACL2

A Computational Logic
for Applicative Common Lisp

ACL2 is both a programming language (Lisp subset) and a logic for theorem proving, developed at UT.

Authors:

- Robert S. Boyer
- J Strother Moore
- Matt Kaufmann
- Warren Hunt

Rewrite Rules

Rewrite Rules express the fact that two things are equal, so one can be rewritten as the other.

```
(defthm plus-commutes
  (equal (+ x y) (+ y x)))

(defthm append-nil
  (implies (true-listp x)
           (equal (append x nil) x)))

(defthm member-append
  (implies
   (and
    (true-listp x)
    (true-listp y))
   (iff (member e (append x y))
        (or (member e x) (member e y)))))
```

Rewrite rules also include heuristics and loop-stopper conditions such as a limit on backchaining.

Proof Techniques

ACL2 uses a variety of proof techniques:

- Rewriting
- Simplification: `(true-listp '()) = true`
- Partial Evaluation: `(and true q) = q`
- Induction
- Backchaining

Uses of ACL2

ACL2 can be used to prove that a hardware implementation meets a specification, e.g. that the AMD X86 chip does in fact implement the X86 instruction set (for which they have a formal specification).

For example, the result of a floating divide instruction is correct according to the IEEE floating point specification.