Efficient Animation of ACL2 Models

Jared Davis
Centaur Technology
jared@centtech.com

ACL2 Seminar
September 30, 2009

Motivation

Applications

- Validating specifications (cosimulation)
- Everyday software in ACL2
- Proof by exhaustive case analysis
- Meta-reasoning and reflection

Timeliness

- Increasing uses of meta-reasoning
  - GL, clause processors, Open ACL2
- Hardware trends (multicore)

Current Approaches

Guards, MBE

- Execution of partial Common Lisp primitives
- Type declarations

Implementation features

- Tail call optimization
- Lisp implementation compiler macros (coerce)

Inlining (macros, defabbrev, misc/definline)

Laziness (macros + MBE)

Current Approaches (2)

Parallelism extension

Hons, memoization

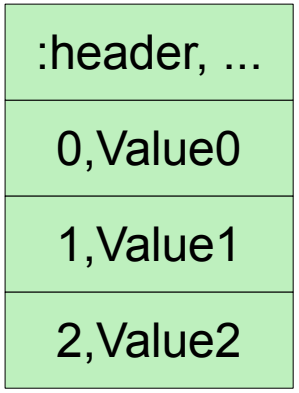
Single-threaded objects

Discipline-based approaches

- ACL2 arrays
- Fast association lists

Review of ACL2 Arrays

In the Logic, an ACL2 array is just a funny alist



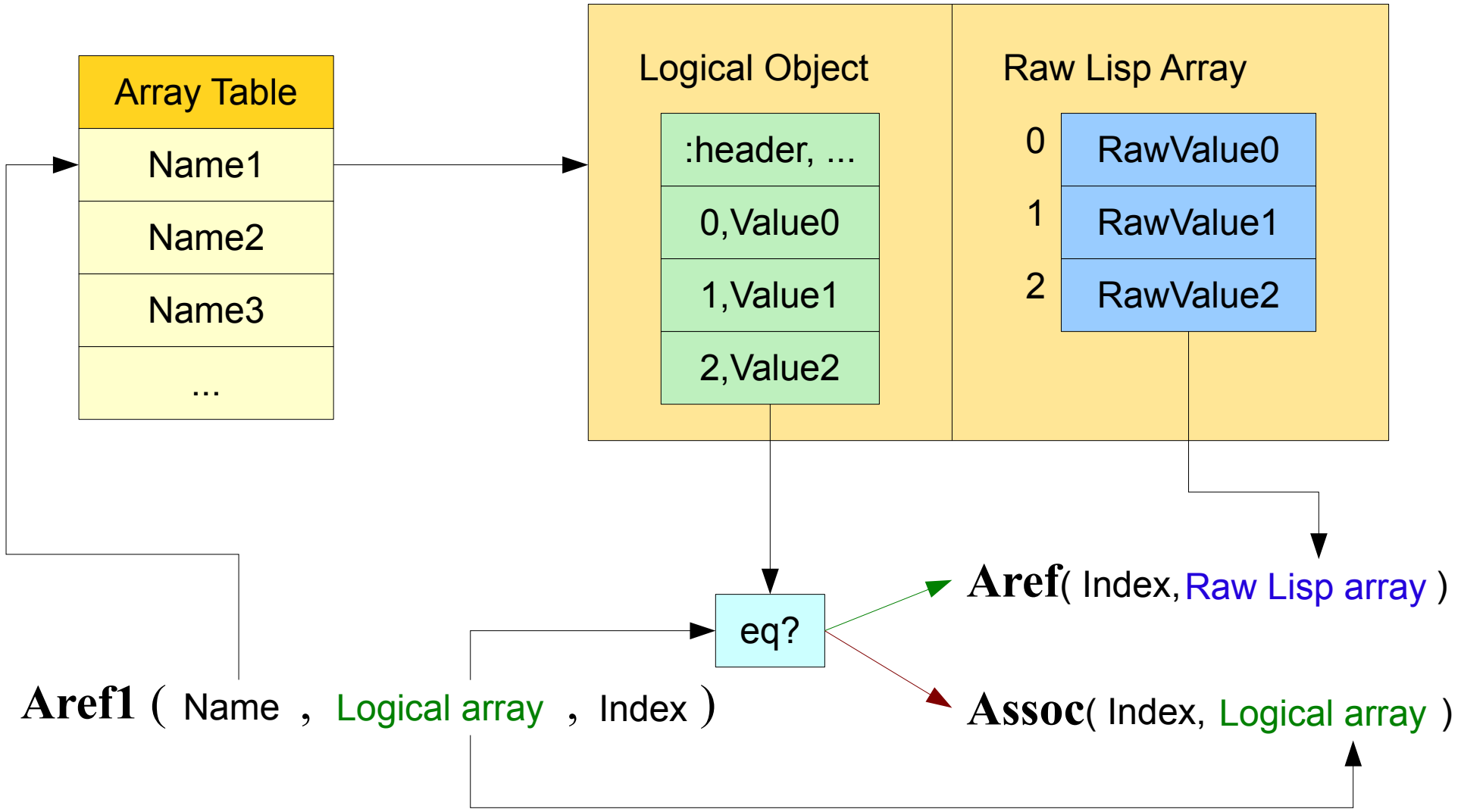
aset1 – like acons

compress1 – removes shadowed pairs

aref1 – like assoc

Header contains a name, size, default value, etc.

Implementation of ACL2 Arrays



Q: Why build the alist?

A: The user might CAR or CDR it.

An Idea

Define our own primitives

Disadvantages

Potentially hard to implement

- ACL2 system code – what has to change?
- Guard system, type declarations

Some overhead

- ACL2::CAR = Wrapper for CL-USER::CAR
- etc.

1. A Trick with Stobjs

2. Experiments with New Primitives

1. Basic approach, read-only arrays

2. Read/write arrays with versioned pointers

3. Read/write arrays with bottom