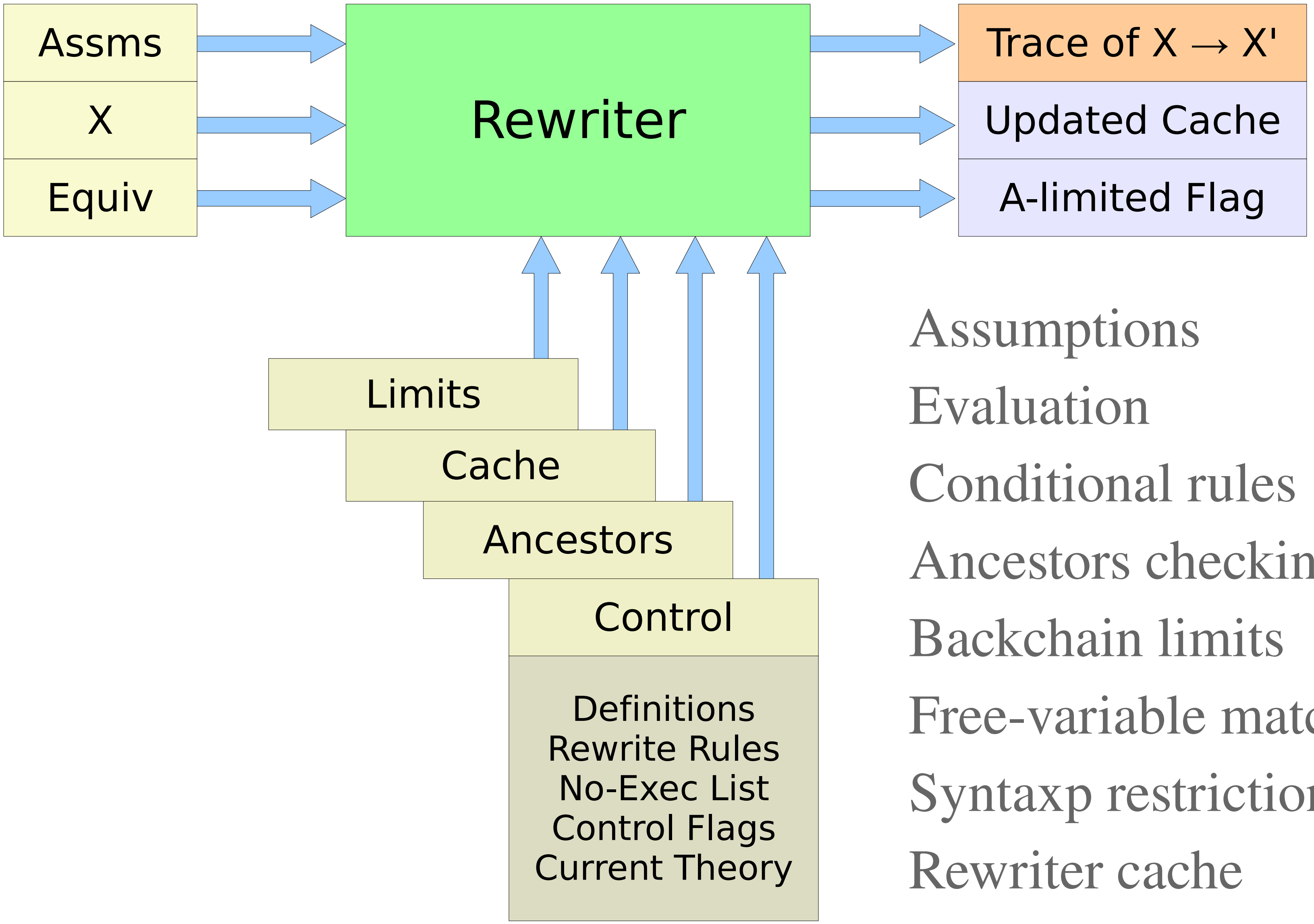
The Milawa Rewriter and an ACL2 Proof of its Soundness

Jared Davis
The University of Texas at Austin
Department of Computer Sciences
jared@cs.utexas.edu

ACL2 ’07

The Milawa Rewriter



Assumptions

Evaluation

Conditional rules

Ancestors checking

Backchain limits

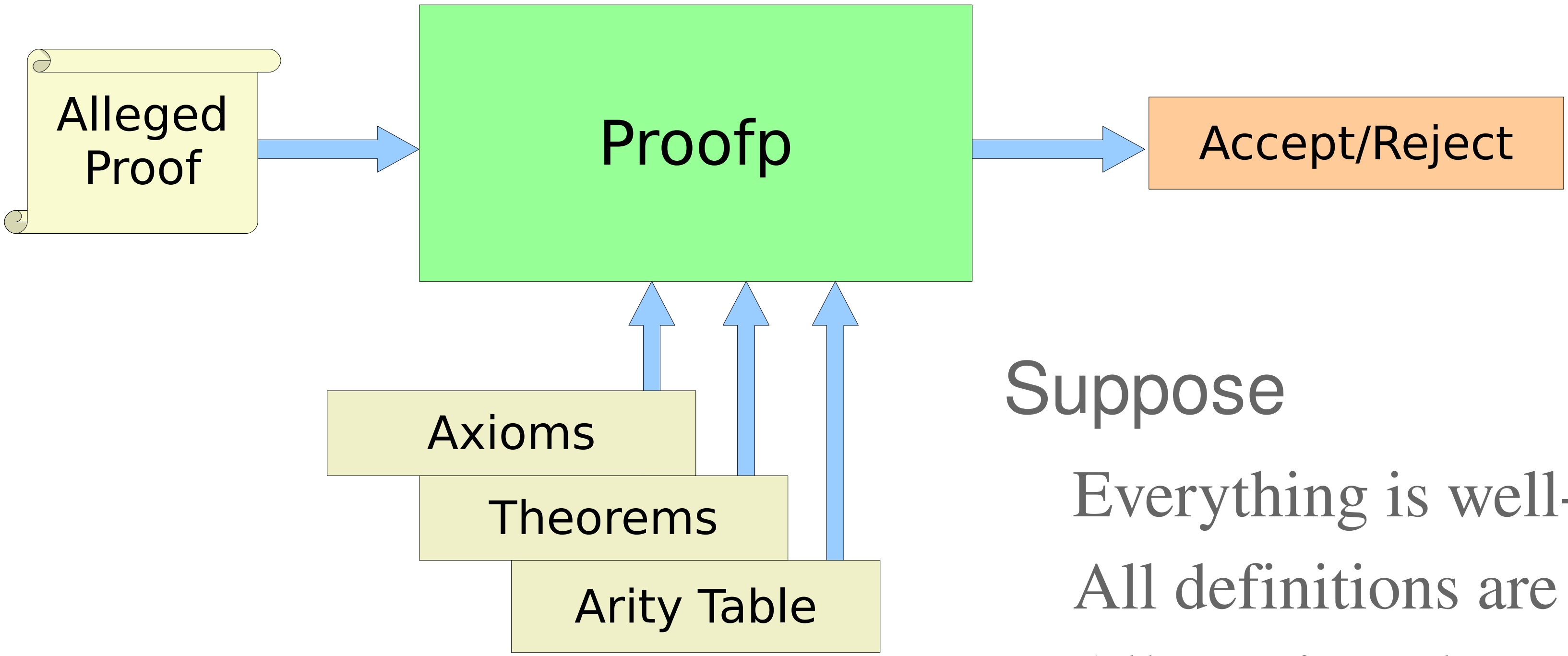
Free-variable matching

Syntaxp restrictions

Rewriter cache

Forcing

Soundness of the Rewriter



Suppose

- Everything is well-formed
- All definitions are axioms
- All rewrite rules are theorems
- X rewrites to X'

Then

- assms → X ≡ X' is provable

The Milawa Logic

Prop. Schema

$$\frac{}{\neg A \vee A}$$

Contraction

$$\frac{A \vee A}{A}$$

Expansion

$$\frac{A}{B \vee A}$$

Associativity

$$\frac{A \vee (B \vee C)}{(A \vee B) \vee C}$$

Cut

$$\frac{A \vee B \quad \neg A \vee C}{B \vee C}$$

Instantiation

$$\frac{A}{A/\sigma}$$

Induction

Reflexivity Axiom

$$x = x$$

Equality Axiom

$$x_1 = y_1 \rightarrow x_2 = y_2 \rightarrow x_1 = x_2 \rightarrow y_1 = y_2$$

Referential Transparency

$$x_1 = y_1 \rightarrow \ldots \rightarrow x_n = y_n \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Beta Reduction

$$((\lambda\, x_1 \ldots x_n \,.\, \beta)\; t_1 \ldots t_n) = \beta/[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$$

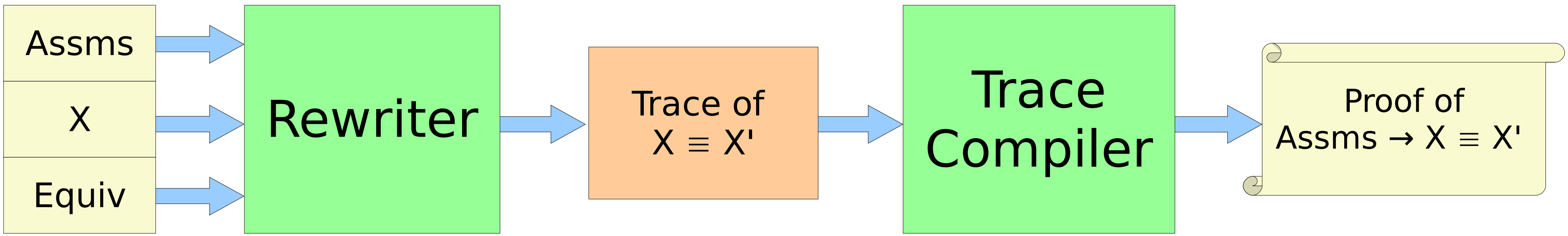
Base Evaluation

e.g., $1+2 = 3$

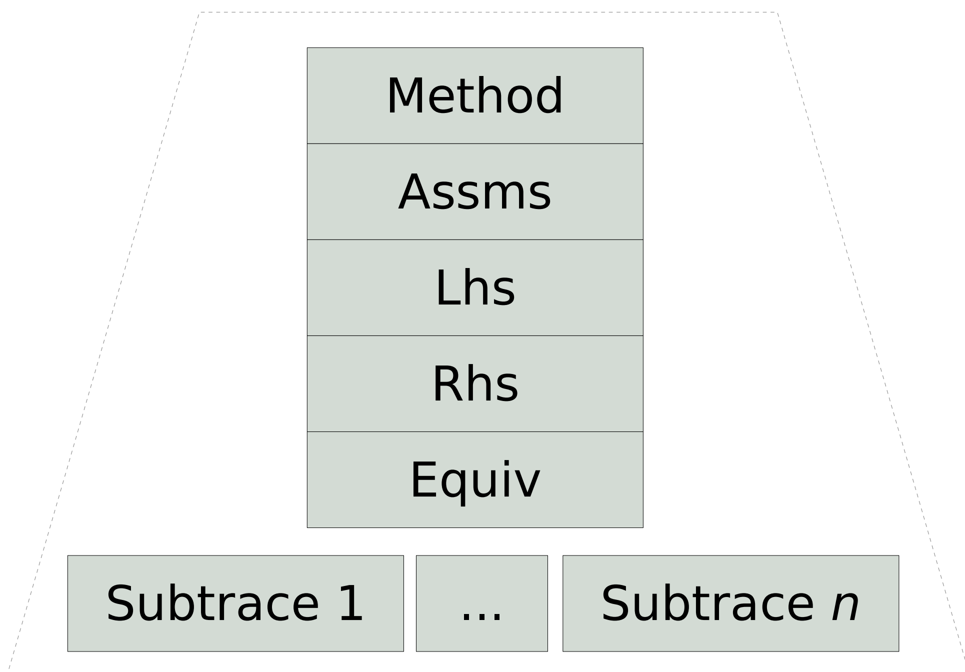
Lisp Axioms

e.g., $consp(cons(x, y)) = t$

Structure of the Proof



Traces



Example: transitivity traces

$$\frac{[\text{assms} \rightarrow]\ X \equiv Y \qquad [\text{assms} \rightarrow]\ Y \equiv Z}{[\text{assms} \rightarrow]\ X \equiv Z}$$

Kinds of Traces

Failure

$$\frac{}{[assms \longrightarrow]\ x \equiv x}$$

If, false case

$$\frac{\begin{array}{l}[assms \longrightarrow]\ x_1 \textit{ iff } \textit{false} \\ [assms \longrightarrow]\ z_1 \equiv z_2\end{array}}{[assms \longrightarrow]\ \textit{if}(x_1, y_1, z_1) \equiv z_2}$$

If, true case

$$\frac{\begin{array}{l}[assms \longrightarrow]\ x_1 \textit{ iff } \textit{true} \\ [assms \longrightarrow]\ y_1 \equiv y_2\end{array}}{[assms \longrightarrow]\ \textit{if}(x_1, y_1, z_1) \equiv y_2}$$

Not congruence

$$\frac{[assms \longrightarrow]\ x \textit{ iff } x'}{[assms \longrightarrow]\ \textit{not}(x) \equiv \textit{not}(x')}$$

Equiv by args

$$\frac{\begin{array}{l}[assms \longrightarrow]\ a_1 = a_1{}' \\ \ldots \\ [assms \longrightarrow]\ a_n = a_n{}'\end{array}}{[assms \longrightarrow]\ f(a_1, \ldots, a_n) \equiv f(a_1{}', \ldots, a_n{}')}$$

Transitivity

$$\frac{\begin{array}{l}[assms \longrightarrow]\ x \equiv y \\ [assms \longrightarrow]\ y \equiv z\end{array}}{[assms \longrightarrow]\ x \equiv z}$$

If, same case

$$\frac{\begin{array}{l}[assms \longrightarrow]\ x_1 \textit{ iff } x_2 \\ x_2, assms \longrightarrow y \equiv w \\ \neg x_2, assms \longrightarrow z \equiv w\end{array}}{[assms \longrightarrow]\ \textit{if}(x_1, y, z) \equiv w}$$

If, general case

$$\frac{\begin{array}{l}[assms \longrightarrow]\ x_1 \textit{ iff } x_2 \\ x_2, assms \longrightarrow y_1 \equiv y_2 \\ \neg x_2, assms \longrightarrow z_1 \equiv z_2\end{array}}{[assms \longrightarrow]\ \textit{if}(x_1, y_1, z_1) \equiv \textit{if}(x_2, y_2, z_2)}$$

If-not normalization

$$\frac{}{[assms \longrightarrow]\ \textit{if}(x, \textit{false}, \textit{true}) \equiv \textit{not}(x)}$$

Lambda equiv by args

$$\frac{\begin{array}{l}[assms \longrightarrow]\ a_1 = a_1{}' \\ \ldots \\ [assms \longrightarrow]\ a_n = a_n{}'\end{array}}{[assms \longrightarrow]\ (\lambda x_1 \ldots x_n \,.\, \beta)\ a_1 \ldots a_n \equiv (\lambda x_1 \ldots x_n \,.\, \beta)\ a_1{}' \ldots a_n{}'}$$

Beta reduction

$$\frac{}{[assms \longrightarrow]\ (\lambda x_1 \ldots x_n \,.\, \beta)\ a_1 \ldots a_n \equiv \beta/[x_1 \leftarrow a_1, \ldots, x_n \leftarrow a_n]}$$

Ground evaluation
(Where *lhs* evaluates to *rhs*)

$$\frac{}{[assms \longrightarrow]\ \textit{lhs} \equiv \textit{rhs}}$$

Rule application
(Justified by a rewrite rule)

$$\frac{\begin{array}{l}[assms \longrightarrow]\ \textit{hyp}_1 \textit{ iff } \textit{true} \\ \ldots \\ [assms \longrightarrow]\ \textit{hyp}_n \textit{ iff } \textit{true}\end{array}}{[assms \longrightarrow]\ \textit{lhs} \equiv \textit{rhs}}$$

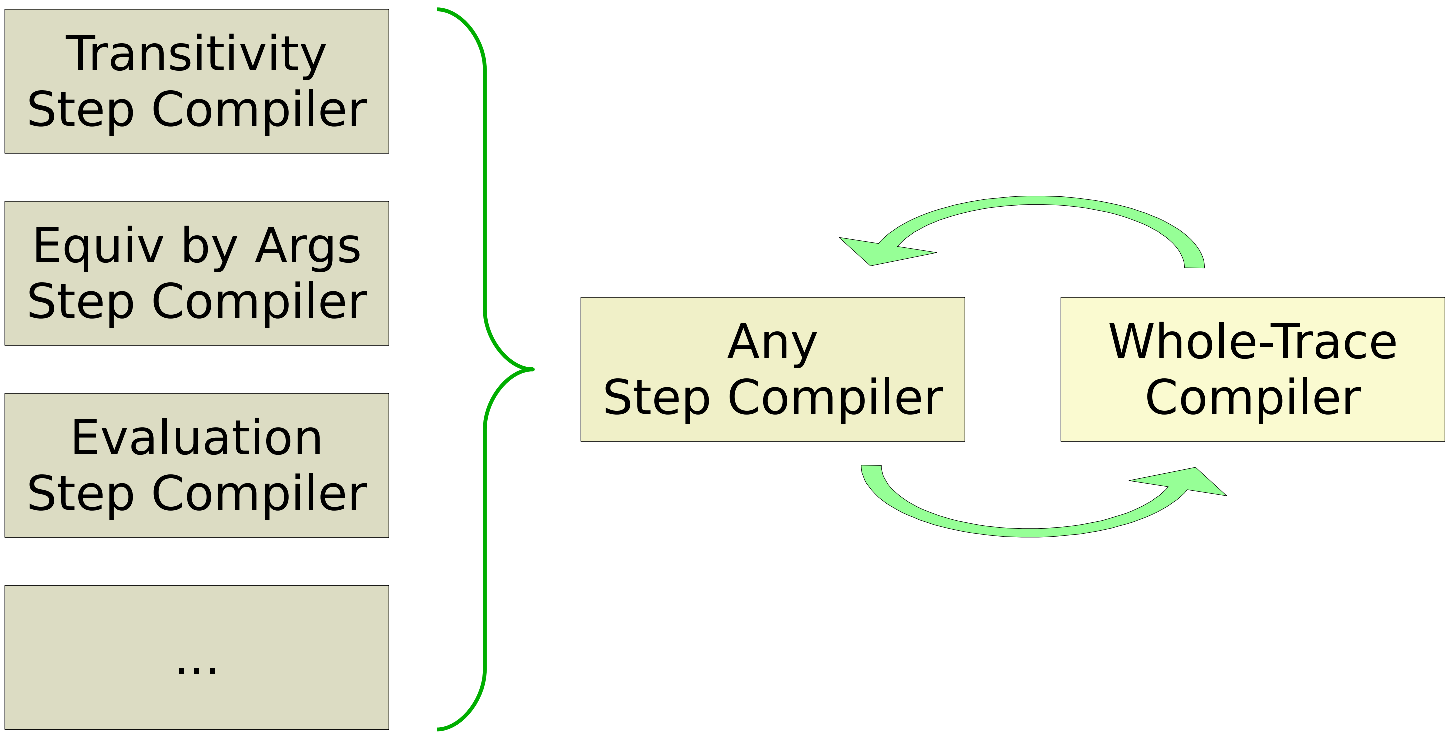
Assumptions
(Justified by an assumption)

$$\frac{}{[assms \longrightarrow]\ \textit{lhs} \equiv \textit{rhs}}$$

Forcing
(Must be justified later)

$$\frac{}{[assms \longrightarrow]\ \textit{lhs} \text{ iff } \textit{true}}$$

Compiling Traces



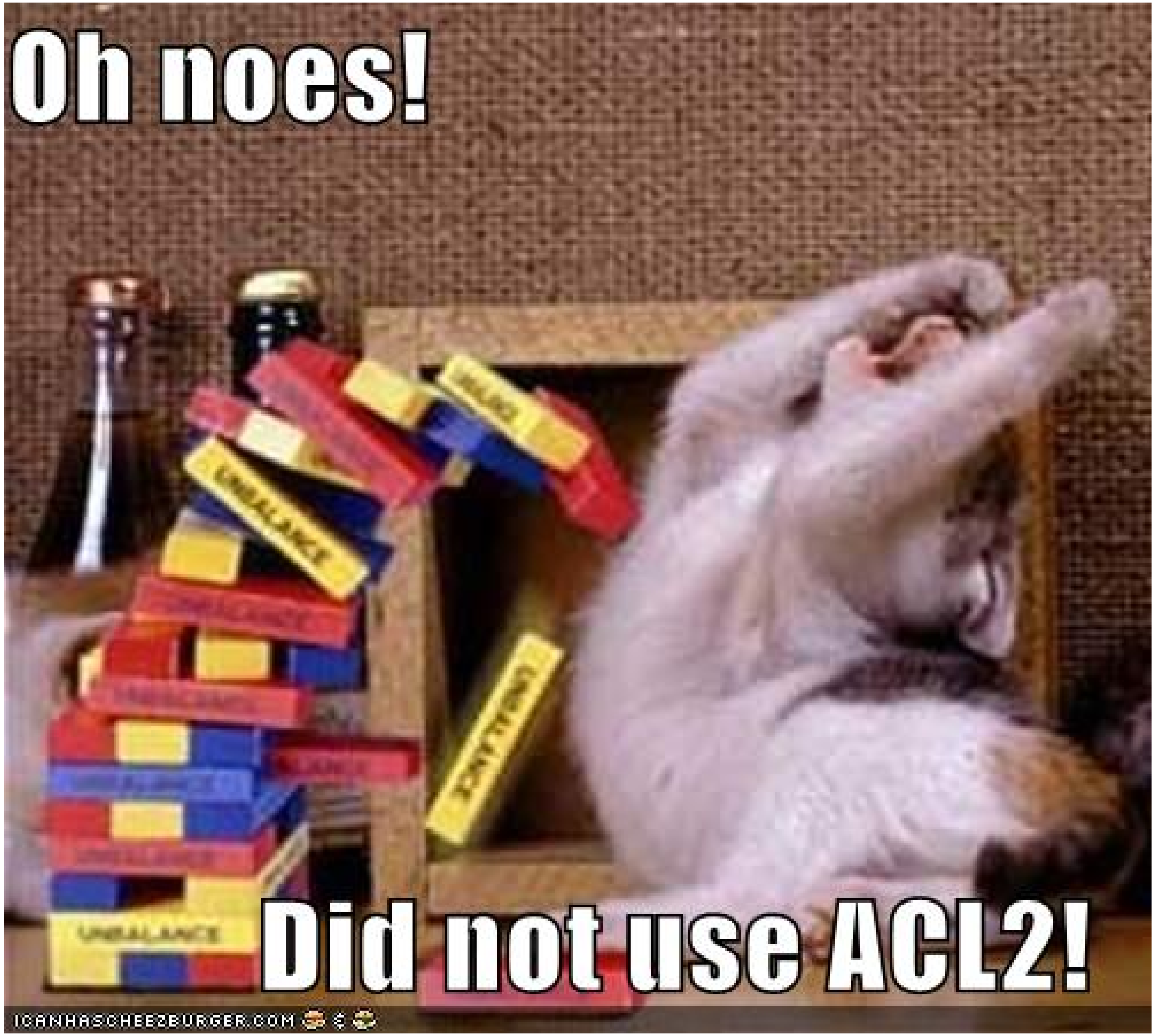
Application to “bootstrapping”

Shameless Plug

The paper is available on my web site

http://www.cs.utexas.edu/~jared

- Defining provability
- The assumptions system
- The evaluator
- Rewrite traces
- The rewriter
- Ancestors checking
- Free-variable matching
- Syntactic restrictions
- Caching
- Forcing


Oh noes!
Did not use ACL2!
UNBALANCE
ICANHASCHEEZBURGER.COM