A Trustworthy, Extensible Theorem Prover

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

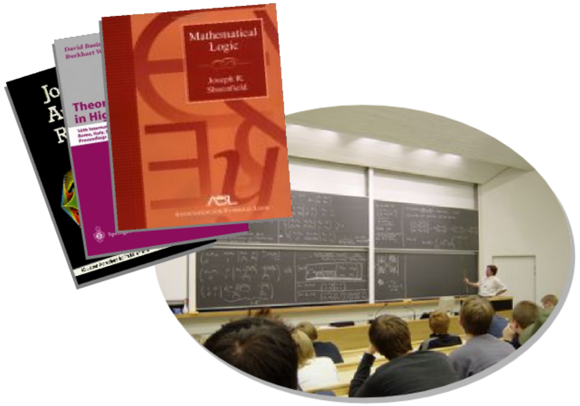
Formal verification

Programs have precise semantics – can be analyzed mathematically

Hoa69

The *social process* does not work

> *DLP77*
>
> "the proofs of even very simple programs run into dozens of pages"



Instead, we use software to build and check *formal proofs*

Mac01

How can we trust this software?

Current approaches

Ad-hoc systems – informal, pragmatic notions of proof

BM81, BM88, GH98, KM98, ORSSC98

LCF style – fully-expansive, space-efficient, safe theorem objects

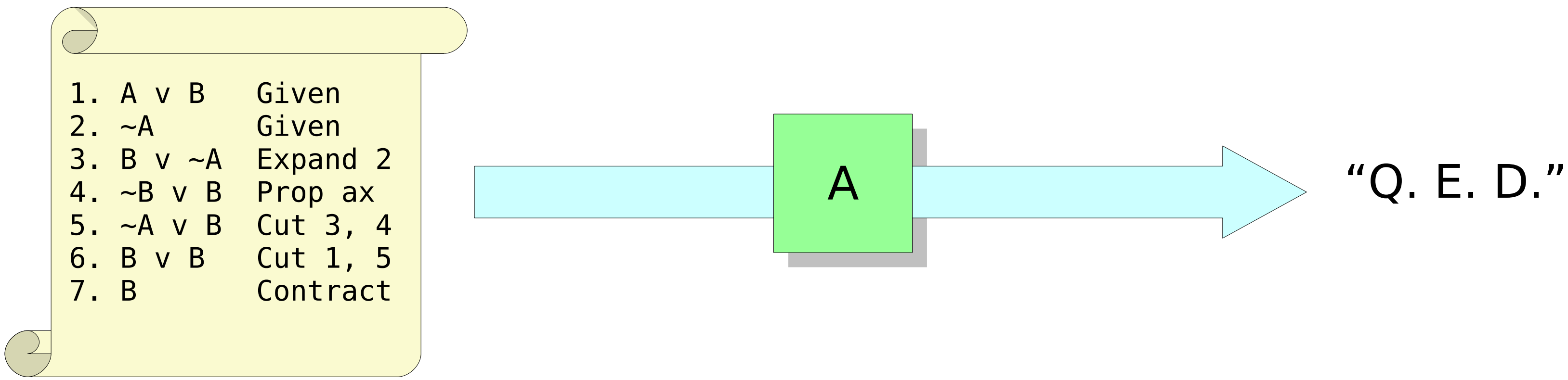
BOU93, HAR95, GOR00, CN05

Constructive type theory – propositions as types, proofs as objects

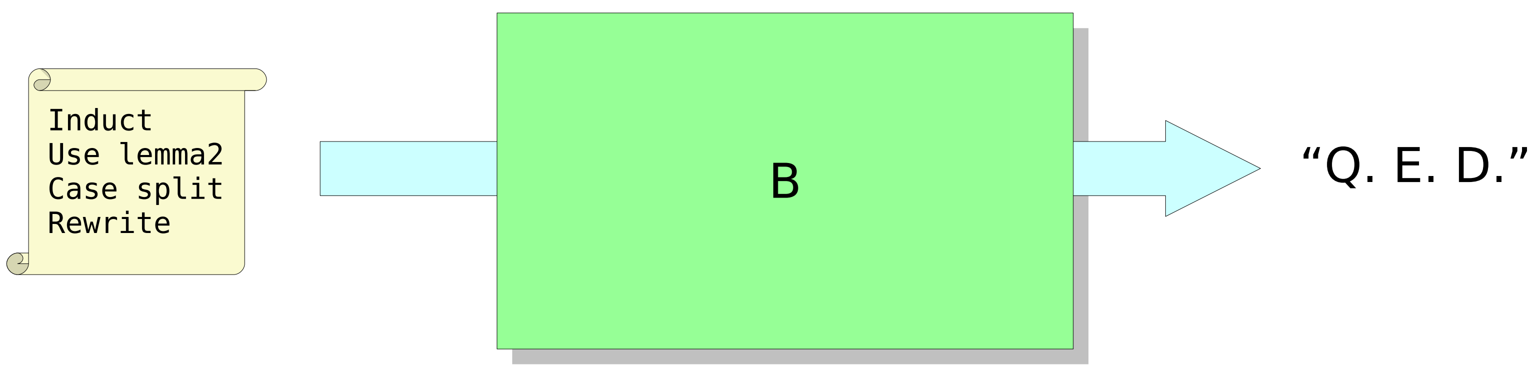
TPG95, Zam97, BC04

A mechanically-verified theorem prover

A is a simple proof checker



B is an automated theorem prover



Construct an *A*-style proof that *B* is sound

Check the proof with *A*

Our language and logic

A pure Lisp

Objects

Naturals
Symbols
Ordered pairs

+

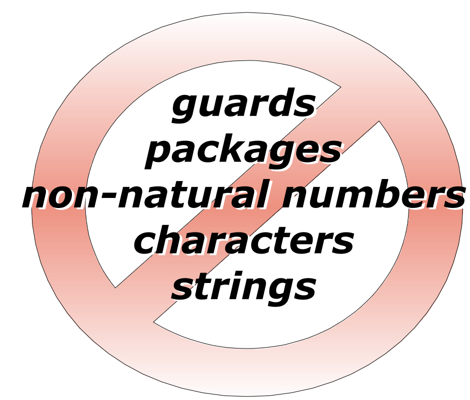
Lisp primitives

if, equal
consp, cons, car, cdr
natp, +, -, <
symbolp, symbol-<

+

Terminating, recursive functions

A simplified ACL2 logic



Our proof checker, *A*, can *see itself* and can *reason about itself*

Our logic at a glance

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

Sho67, KM98

Our proof checker, A

List utilities
len, app, rev, memberp, uniquep, ...

63 lines
11 functions

Terms and formulas
recognizers, constructors, accessors

163 lines
39 functions

Substitution
substitutions, applying substitutions

62 lines
8 functions

Proof encoding
recognizer, accessors

27 lines
8 functions

Proof checking
step checkers, whole-proof checking

325 lines
27 functions

640 lines, 93 functions

+

Command line program

Lisp package	59 lines
Primitives	95 lines
File reader	108 lines
Termination	106 lines
Events and state	82 lines
Translation	192 lines
Initial axioms	113 lines

755 lines of Common Lisp

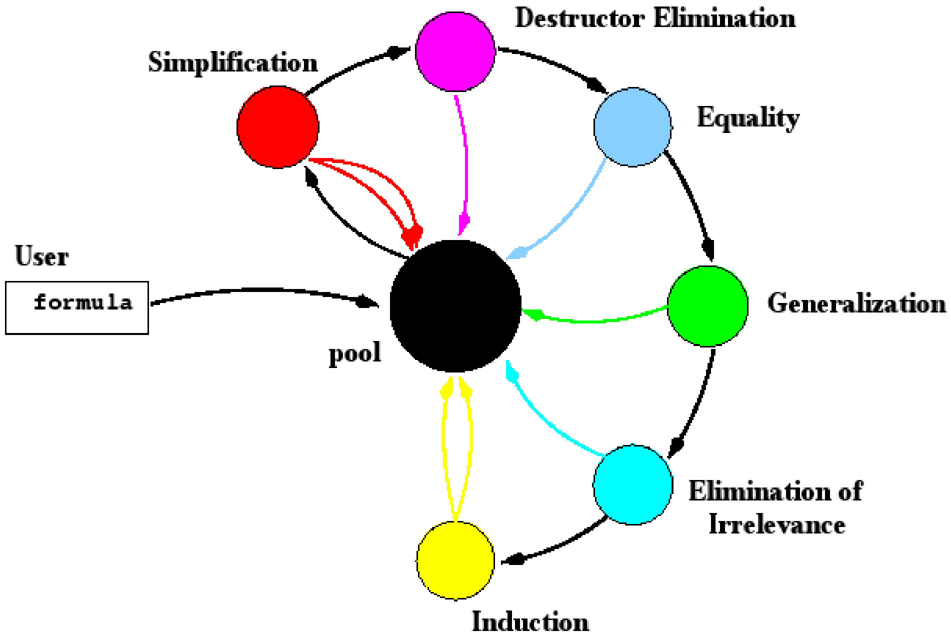
+

Lisp environment
Allegro, CMUCL, OpenMCL, ...

Avi95, Mac01

Our theorem prover, B

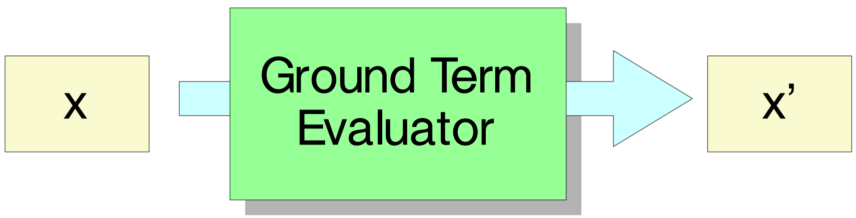
Styled after ACL2



Written in our logic, designed for verification

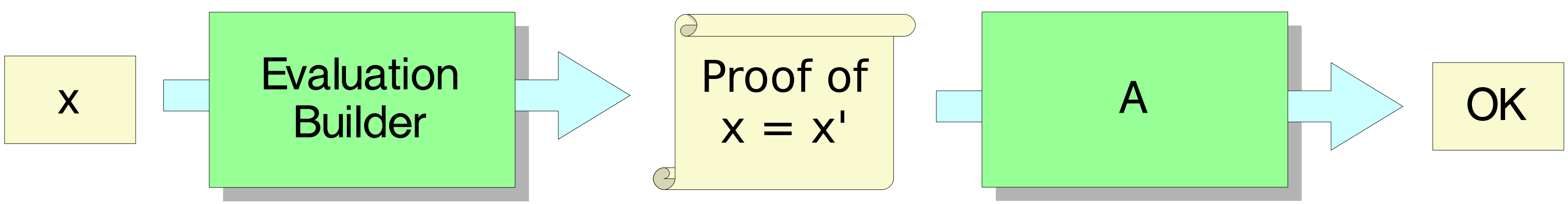
Planning the proof of B's soundness

Sketching the proofs with "ACL2-lite" –
translate into A-style proofs later



Soundness claim
x = x' is provable

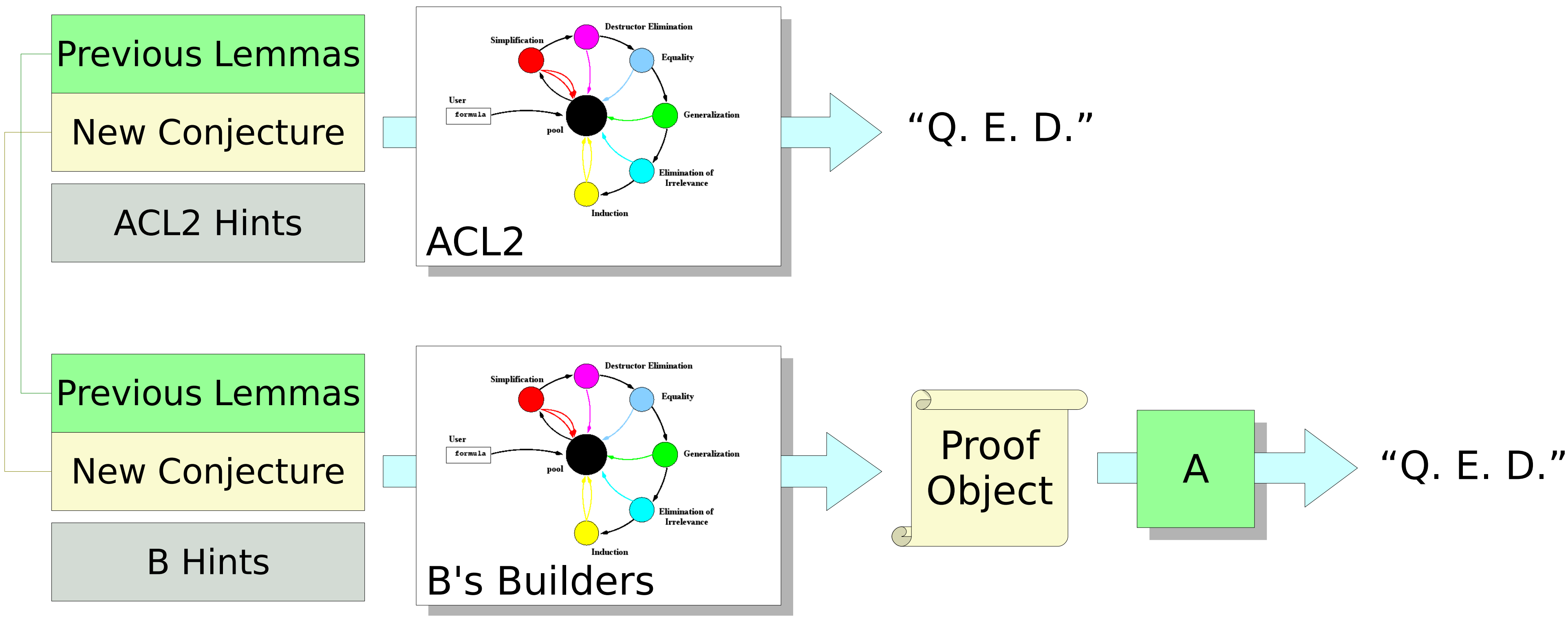
Proving the soundness claims



Net result: ACL2 lemma libraries

Translating the lemma libraries

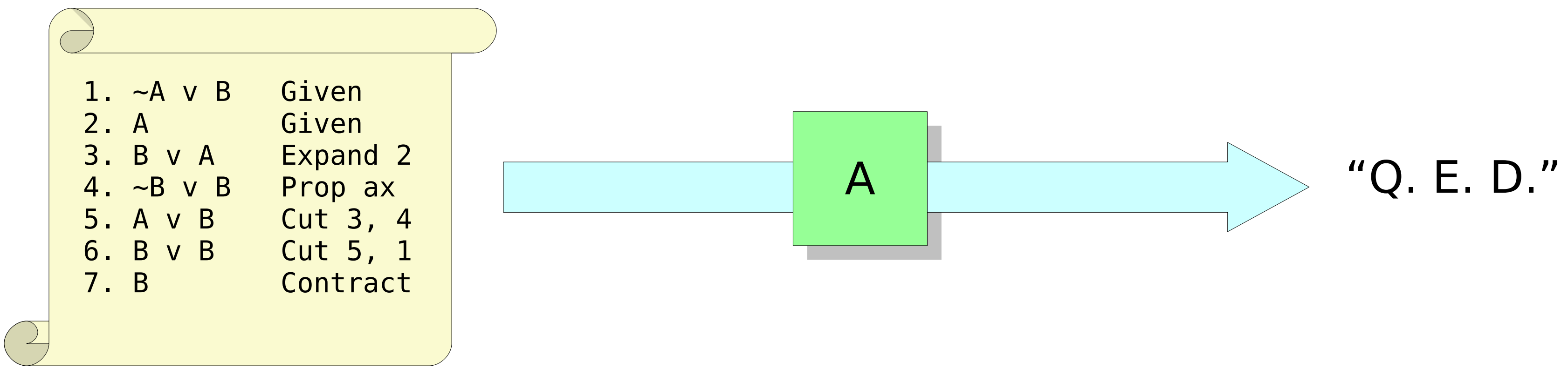
Use B (and its builders) to replay lemmas

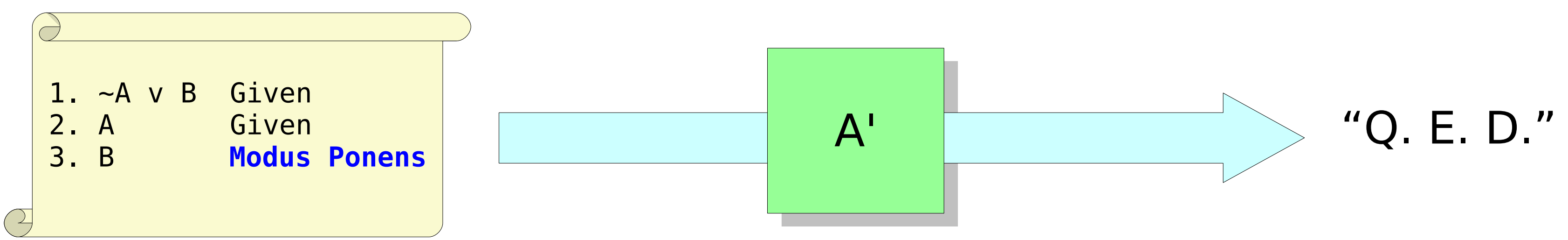


Proof size must be carefully managed

A stack of verified proof checkers

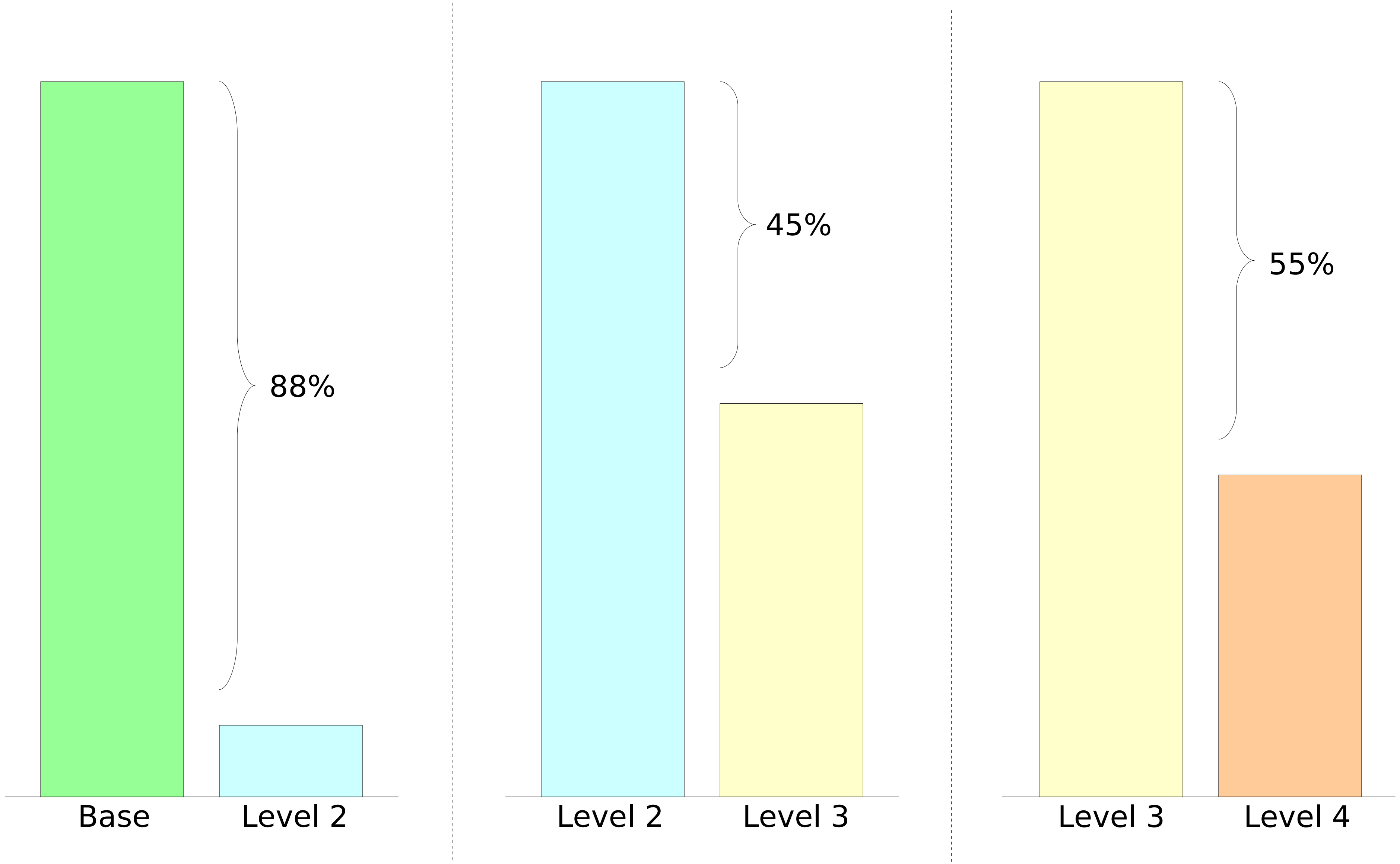
Use A to verify A', A' to verify A'', ..., until we get to B





We now have three verified checkers

Significance of proof size reductions



Present work

Implemented A and its command loop

Wrote B and verified its proof methods with "ACL2-lite"

Translated 4,500 lemmas, including three extended proof checkers

Contributions

Metatheory as an approach to building practical theorem provers

Verified theorem proving algorithms

Highly-extensible proof construction

Efficient proof construction through verified proof methods

Potential target for other systems

Related work

Embedding proof checkers in a logic

Göd31, Sha94

Mechanically-verified proof checkers

vW94, RM05, Har06

Independent proof checking

MS00, CC02, OS06

Metafunctions

Encoding terms, defining evaluators and metafunctions, soundness, integration

metafunction

meaning(fn(term), env) = meaning(term, env)

evaluator

encoded term

Support for metafunctions

BM81, KC86, Sli92, SNG+04, CN05, GM05

References

[Avi95] Algirdas A. Avižienis. The methodology of n-version programming. In *Software Fault Tolerance*. Wiley, 1995.

[BC04] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development: Coq'Art: the Calculus of Inductive Constructions*. 2004.

[BM81] R. S. Boyer and J S. Moore. Metafunctions: proving them correct and using them efficiently as new proof procedures. In *The Correctness Problem in Computer Science.* Academic Press, 1981.

[BM88] R. S. Boyer and J S. Moore. Integrating decision procedures into heuristic theorem provers: a case study of linear arithmetic. In *Machine Intelligence 11.* Oxford University Press, 1998.

[Bou92] R. Boulton. *Efficiency in a Fully-Expansive Theorem Prover*. Ph.D. thesis, University of Cambridge, 1993.

[CC02] J. L. Caldwell and J. Cowles. Representing Nuprl proof objects in ACL2: toward a proof checker for Nuprl. ACL2 '02. 2002.

[CN05] A. Chaieb and T. Nipkow. Verifying and reflecting quantifier elimination for Presburger arithmetic. LPAR '95. 1995.

[DLP77] R. A. DeMillo, R. J. Lipton, and A. J. Perlis. Social processes and proofs of theorems and programs. POPL '77. 1977.

[GH98] D. Griffioen and M. Huisman. A comparison of PVS and Isabelle/HOL. TPHOLS '98. 1998.

References

[GM05] B. Grégoir and A. Mahboubi. Proving equalities in a commutative ring done right in Coq. TPHOLS '05. 2005.

[Göd31] K. Gödel. On formally undecidable propositions of *Principia Mathematica* and related systems I. In S. Feferman, ed., *Kurd Gödel: Collected Works*, volume 1. Oxford University Press, 1986.

[Gor00] M. Gordon. From LCF to HOL: a short history. In *Proof, Language, and Interaction*. MIT Press, 2000.

[Har95] J. Harrison. Metatheory and reflection in theorem proving: a survey and critique. Technical report CRC-053, SRI Cambridge, 1995.

[Har06] J. Harrison. Towards self-verification of HOL Light. IJCAR '06. 2006.

[Hoa69] C. A. R. Hoare. An axiomatic basis for computer programming. In *Communications of the ACM*, 12(10), October 1969.

[KC86] T. B. Knoblock and R. L. Constable. Formalized metareasoning in type theory. LICS '86. 1986.

[KM98] M. Kaufmann and J S. Moore. A precise description of the ACL2 logic, 1998.

[Mac01] D. MacKenzie. *Mechanizing Proof: Computing, Risk, and Trust*. MIT Press, 2001.

[MS00] W. McCune and O. Shumsky. Ivy: a preprocessor and proof checker for first-order logic. In *Computer-Aided Reasoning: ACL2 Case Studies*. Kluwer, 2000.

References

[ORSSC98] S. Owre, J. M. Rushby, N. Shankar, and D. W. J. Stringer-Calvert. PVS: an experience report. In *Applied Formal Methods—FM-Trends 98*. 1998.

[OS06] S. Obua and S. Skalberg. Importing HOL into Isabelle/HOL. IJCAR '06. 2006.

[RM95] T. Ridge and J. Margetson. A mechanically verified, sound and complete theorem prover for first order logic. TPHOLS '05. 2005.

[SNG+04] E. Smith, S. Nelesen, D. Greve, M. Wilding, and R. Richards. An ACL2 library for bags. ACL2 '04. 2004.

[Sha94] N. Shankar. *Metamathematics, Machines, and Gödel's Proof*. Cambridge University Press, 1994.

[Sho67] J. R. Shoenfield. *Mathematical Logic*. Association for Symbolic Logic, 1967.

[Sli92] K. Slind. Adding new rules to an LCF-style logic implementation: preliminary report. TPHOLS '92. 1992.

[TPG95] The PRL Group. Implementing mathematics with the Nuprl proof development system. Cornell University, 1995.

[vW94] J. von Wright. The formal verification of a proof checker. SRI Internal Report, 1994.

[Zam97] V. Zammit. A comparative study of Coq and HOL. TPHOLS '97. 1997.