The Milawa Theorem Prover is Sound

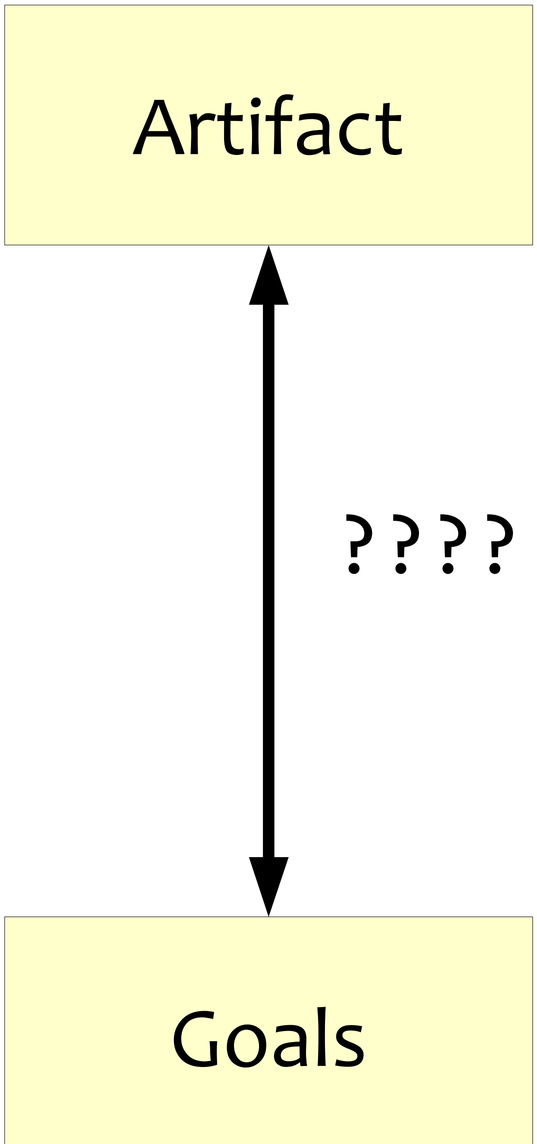
down to the x86 machine code that runs it

Jared Davis
Centaur Technology, USA

Magnus Myreen
University of Cambridge, UK

Slides for Northeastern University, July 2012

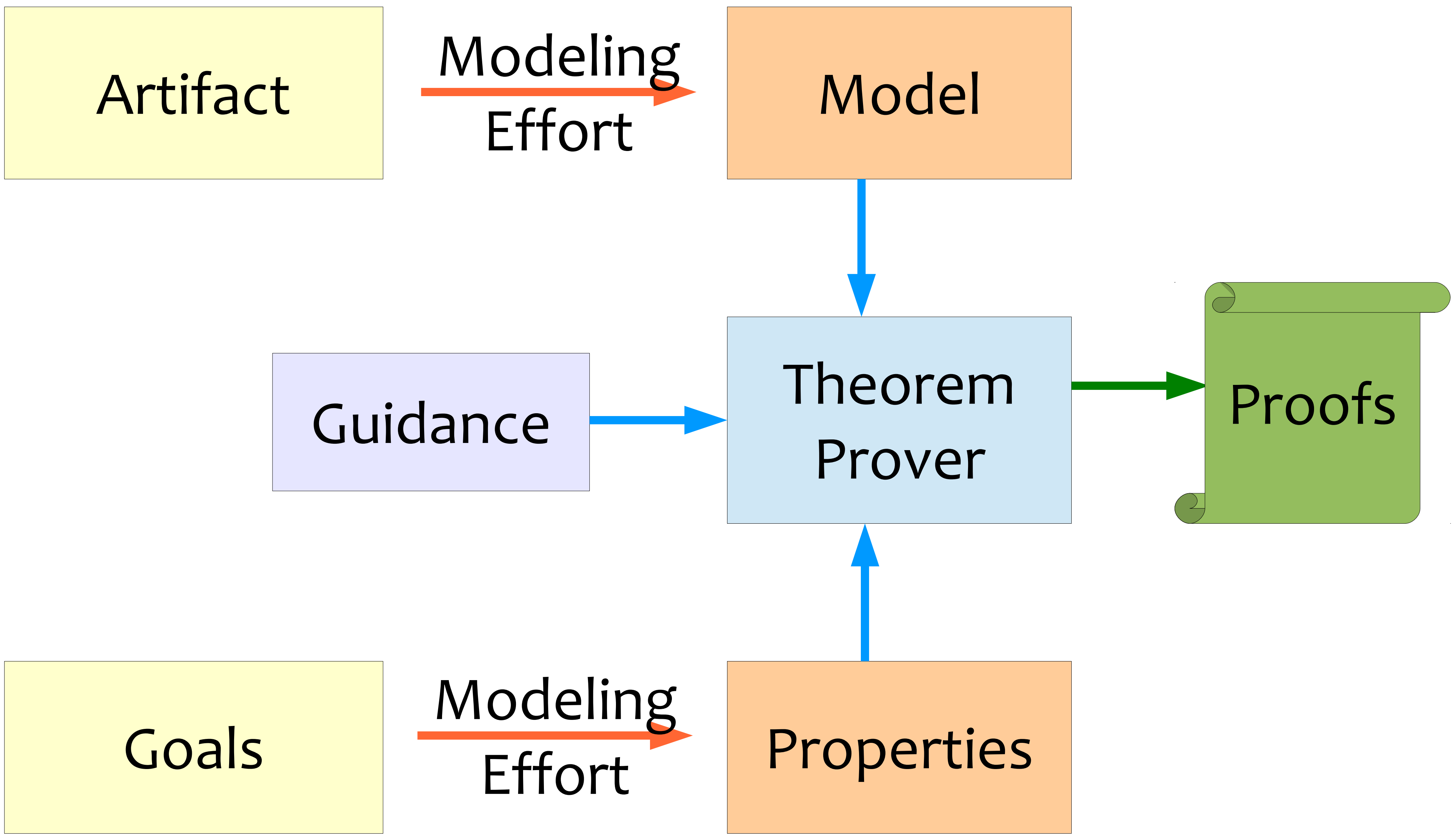
Correctness in Computer Science



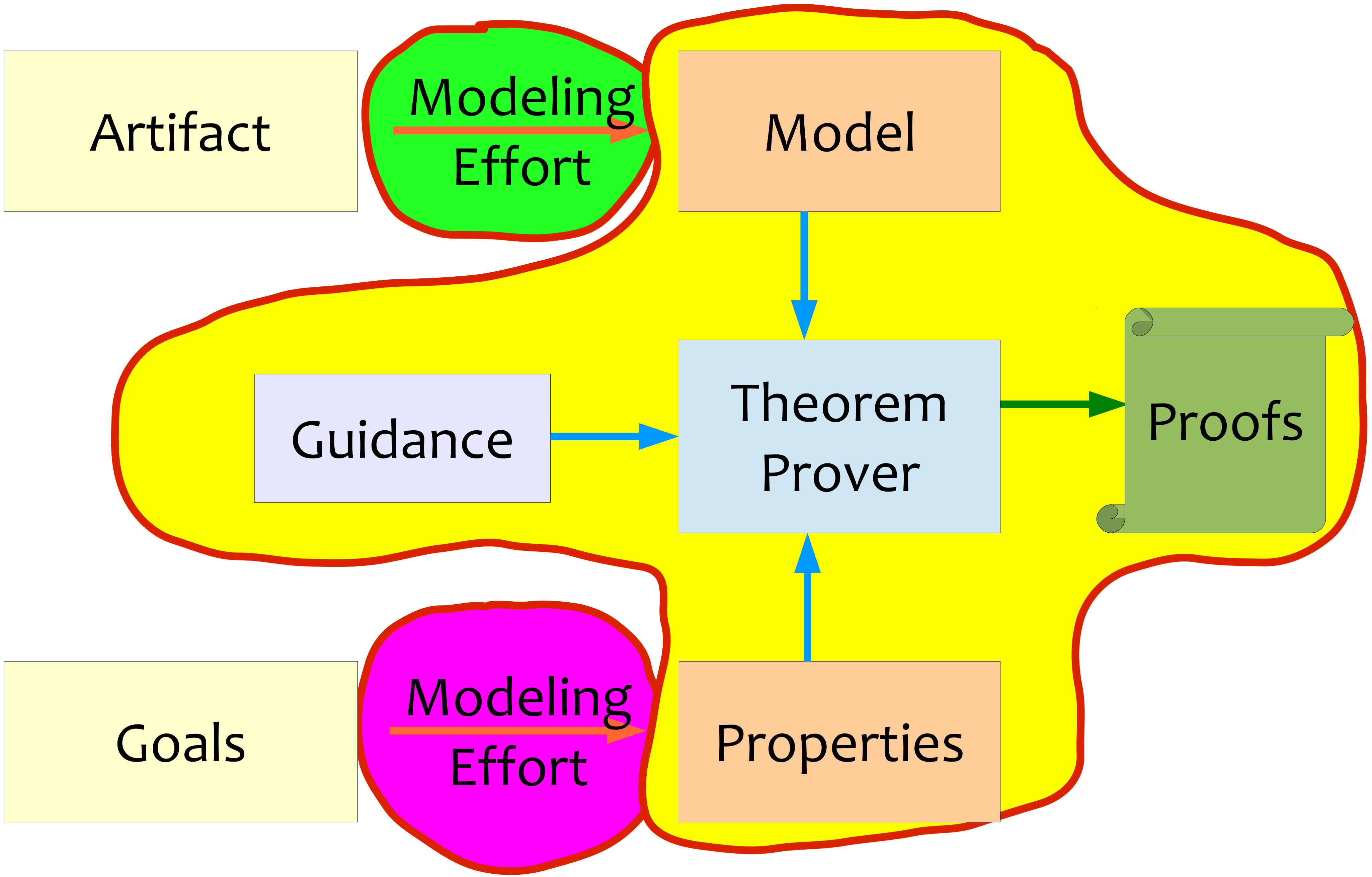
Program
Hardware module
Algorithm
Protocol
Type system

Implements the C standard
Correctly divides floats
Finds the minimum spanning tree
Transactions are secret
Subtyping is transitive

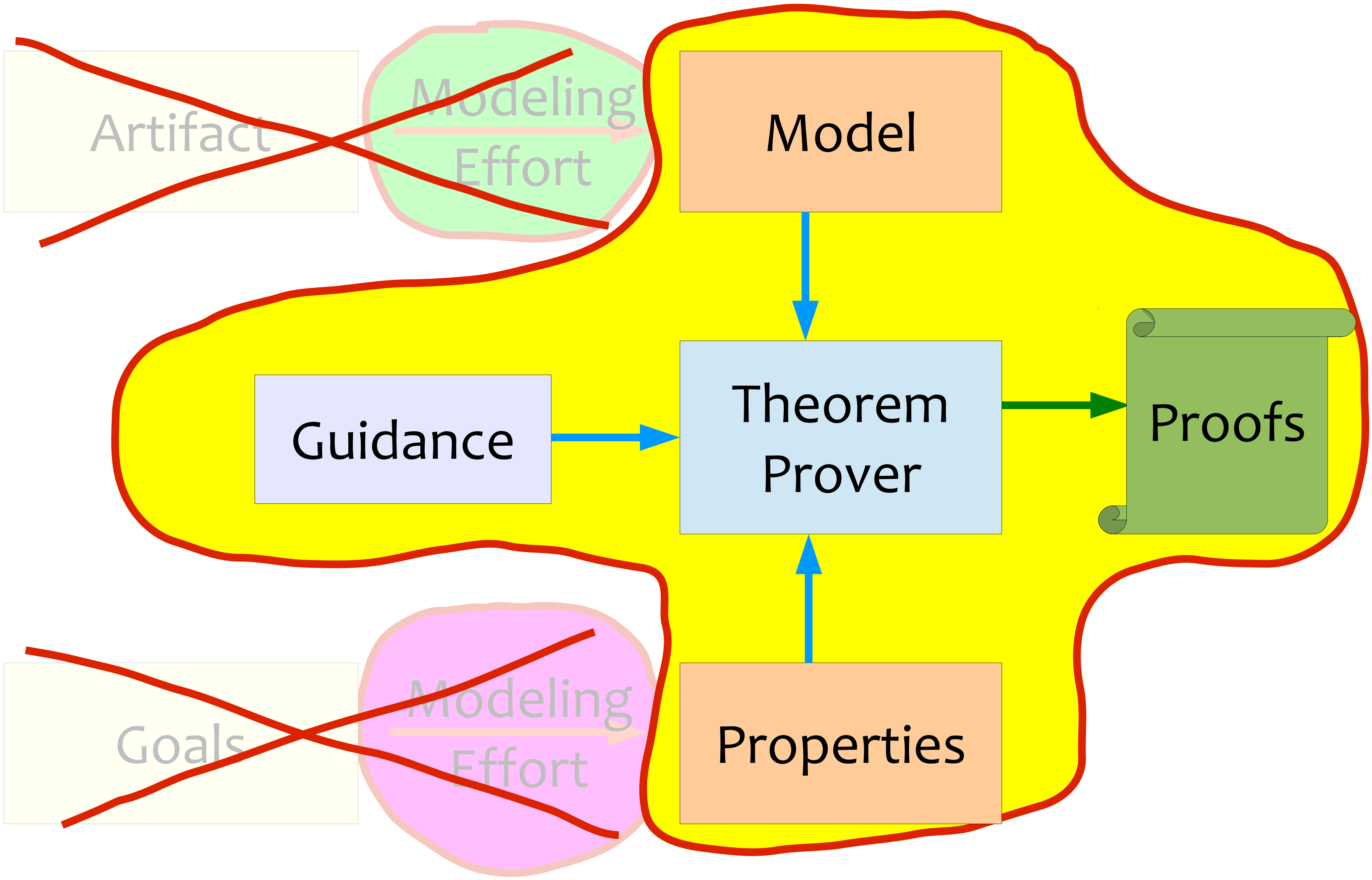
Formal Verification with Theorem Provers



Causes for Concern

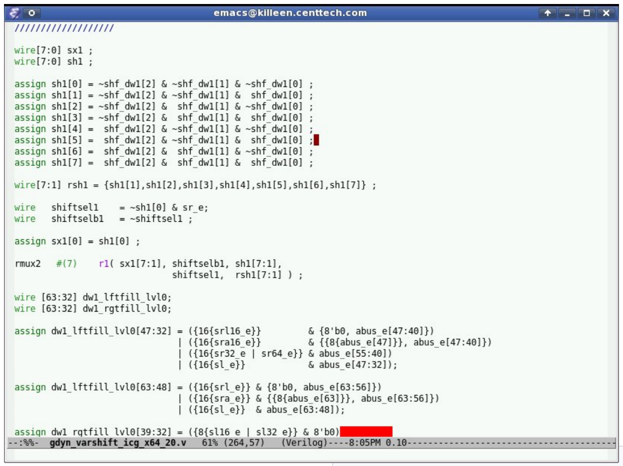


Focus: Soundness of the Theorem Prover



Scope of the Theorem Prover

850K line design



Translator

125K loc

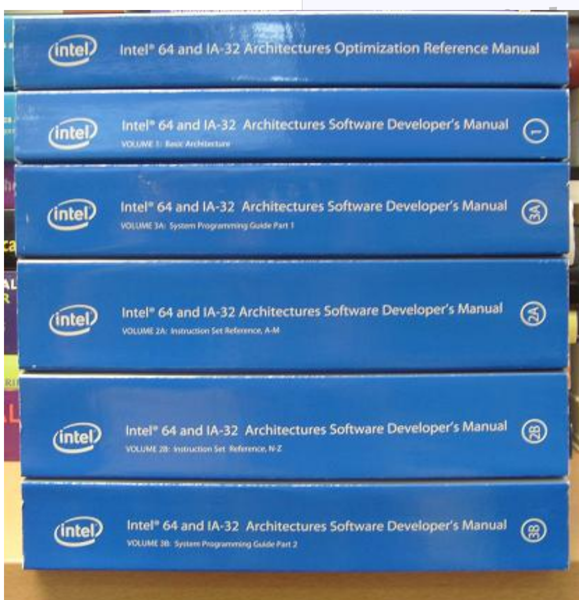
Model

350K loc

ACL2 — A Computational Logic — Applicative Common Lisp

ance

Proofs



Human

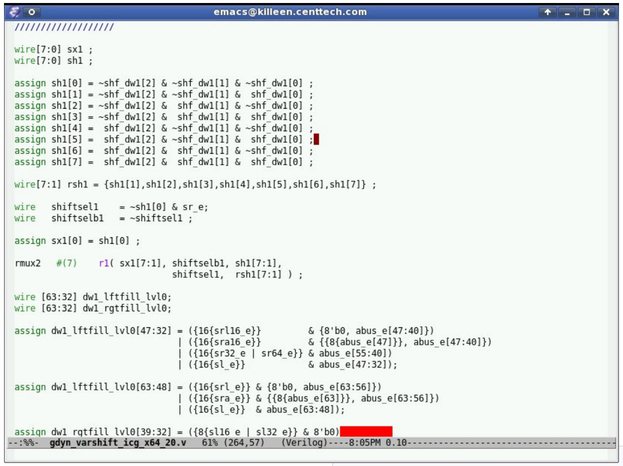
Effort

Properties

Heavy, heavy books

Scope of the Theorem Prover

850K line design



Translator

125K loc

Model

350K loc

300K lisp lines

70K c/asm lines

A COMPUTATIONAL LOGIC

ACL2

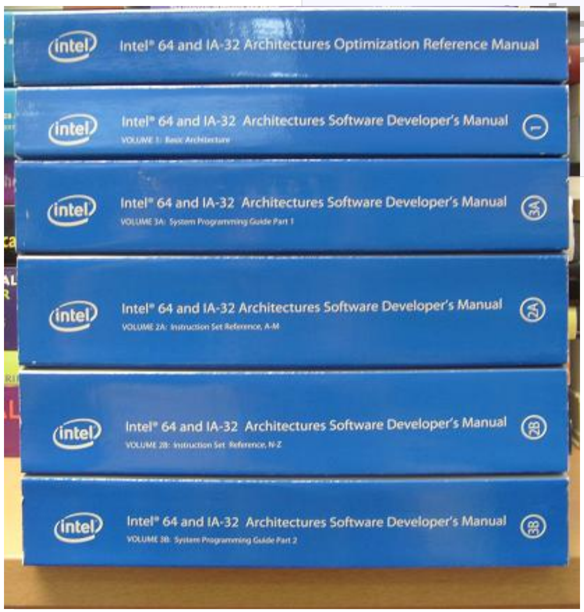
APPLICATIVE COMMON LISP

+ clozure associates

GCC 2-4M lines

6-15M lines

ance

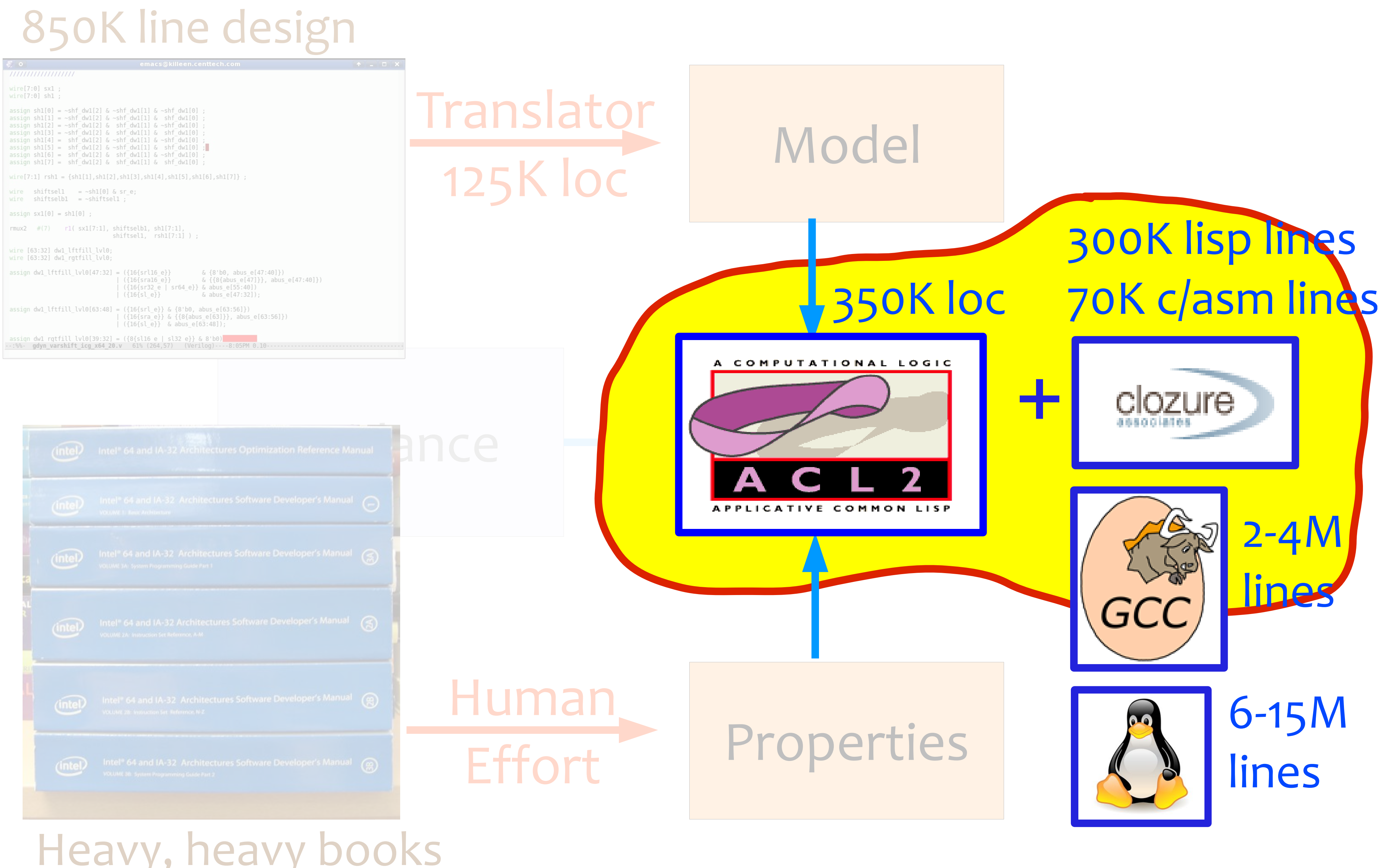


Human Effort

Properties

Heavy, heavy books

This Talk


850K line design
Translator
125K loc
Model
Human
Effort
Properties
Heavy, heavy books
Intel® 64 and IA-32 Architectures Optimization Reference Manual
Intel® 64 and IA-32 Architectures Software Developer's Manual
350K loc
300K lisp lines
70K c/asm lines
A COMPUTATIONAL LOGIC
ACL2
APPLICATIVE COMMON LISP
+
clozure
associates
GCC
2-4M
lines
6-15M
lines

What is Milawa?

Unsound, Unverified

User Interface

Interactive, command-line program

- Define functions
- Propose theorems
- Manage proof attempts

```
MILAWA !> (%defun in (a x)
             (if (consp x)
                 (or (equal a (car x))
                     (in a (cdr x)))
               nil)
             :measure (len x))
```

Termination Proof

```
(%defun in (a x)
   (if (consp x)
       (or (equal a (car x))
           (in a (cdr x)))
     nil)
   :measure (len x))
```

```
Two goals remain.

   1. (EQUAL (ORDP (LEN X)) 'T)

   2. (IF (EQUAL (ORD< (LEN (CDR X)) (LEN X)) 'T)
          'T
          (IF (EQUAL (CONSP X) 'NIL)
              'T
              (EQUAL (NOT (EQUAL A (CAR X))) 'NIL)))

MILAWA !>
```

Termination Proof

```
(%defun in (a x)
   (if (consp x)
       (or (equal a (car x))
           (in a (cdr x)))
     nil)
   :measure (len x))
```

```
MILAWA !>(%split)
; Splitting clause 2.
; Splitting clause 1.
Two goals remain.

 1. (EQUAL (ORDP (LEN X)) 'T)


 2. (IMPLIES (AND (NOT (EQUAL A (CAR X)))
                  (CONSP X))
             (EQUAL (ORD< (LEN (CDR X)) (LEN X))
                    'T))


MILAWA !>
```

```
MILAWA !>(%crewrite default)
; Rewrote clause #2 in 0.001999 seconds (proved) ...
; Rewrote clause #1 in 0.038994 seconds (proved) ...
; Rewrote 2 clauses; 0 (+ 0 forced) remain.
All goals have been proven.
```

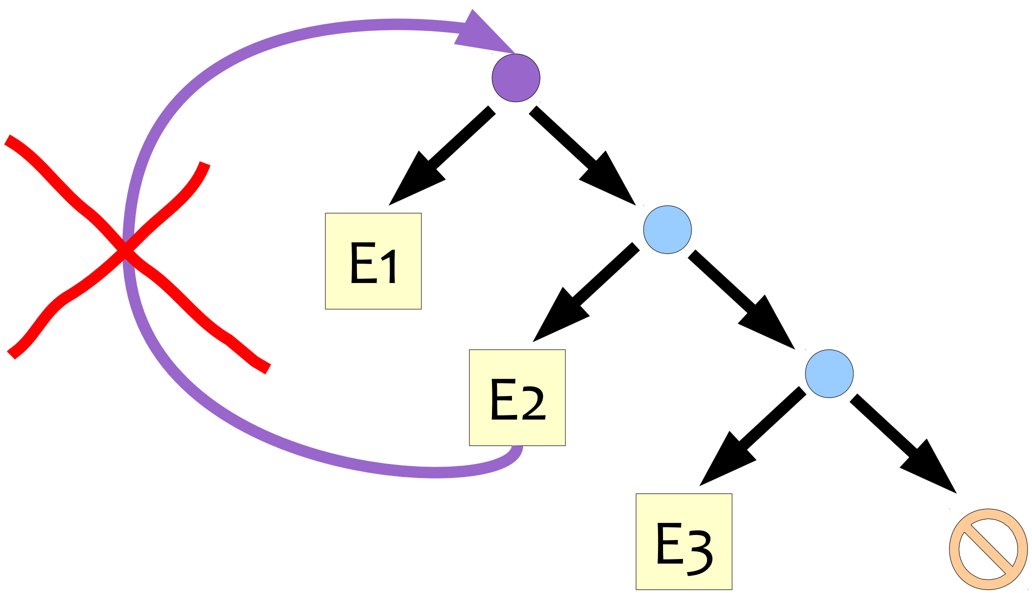
```
MILAWA !>(%crewrite default)
; Rewrote clause #2 in 0.001999 seconds (proved) ...
; Rewrote clause #1 in 0.038994 seconds (proved) ...
; Rewrote 2 clauses; 0 (+ 0 forced) remain.
All goals have been proven.

MILAWA !>(%admit)
; Compiling worlds for IN...
; Compiling proofs for IN...
...
;; Preparing to admit IN.
;; Proof sizes total: 3,409,472 conses ...
; Checking the proofs...
...
; Proof-checking completed.
;; Proofs accepted.  Saving as user/admit-in.proofs
...
New rule: IN

MILAWA !>
```

Example Theorem: (not (in a a))

Let's prove lists can't be in themselves



This can't happen, no circular conses

Lists have to be *bigger* than their elements

The Key Lemma

Lists have to be *bigger* than their elements:

```
MILAWA !>(%defthm rank-when-in
            (implies (in a x)
                     (< (rank a) (rank x))))
One goal remains.

   1. (IMPLIES (AND (IN A X))
               (IFF (< (RANK A) (RANK X)) 'T))
```

```
One goal remains.

   1. (IMPLIES (AND (IN A X))
               (IFF (< (RANK A) (RANK X)) 'T))

MILAWA !>(%cdr-induction x)
... five subgoals ...

MILAWA !>(%auto)
... various progress messages ...
Two goals remain.

   1. (IMPLIES (AND (NOT (CONSP X)))
               (NOT (IN A X)))

   2. (IMPLIES (AND (IN A (CONS X1 X2))
                    (NOT (IN A X2)))
               (< (RANK A)
                  (+ '1 (+ (RANK X1)
                           (RANK X2)))))
```

```
One goal remains.

   1. (IMPLIES (AND (IN A X))
               (IFF (< (RANK A) (RANK X)) 'T))

MILAWA !>(%cdr-induction x)
... five subgoals ...

MILAWA !>(%auto)
... various progress messages ...
Two goals remain.

   1. (IMPLIES (AND (NOT (CONSP X)))
               (NOT (IN A X)))

   2. (IMPLIES (AND (IN A (CONS X1 X2))
                    (NOT (IN A X2)))
               (< (RANK A)
                  (+ '1 (+ (RANK X1)
                           (RANK X2)))))
```

A good rule!

A good rule!

```
MILAWA !>(%qed)
; Compiling worlds for RANK-WHEN-IN...
...
; Preparing to check RANK-WHEN-IN.
;; Proof size: 4,712,680 conses.
; Checking the proof.
...
;; Proof accepted.  Saving as user/thm-rank-when-
in.proof
New rule: RANK-WHEN-IN
```

Our Goal Theorem

```
MILAWA !>(%defthm not-in-self
           (not (in a a)))

One goal remains.

  1. (EQUAL (IN A A) 'NIL)

MILAWA !>(%use (%instance (%thm rank-when-in)
                          (x a)))
... one goal with messy ifs ...
MILAWA !>(%split) ;; to clean it up
One goal remains.

  1. (IMPLIES (AND (IFF (< (RANK A) (RANK A)) 'T))
              (NOT (IN A A)))
```

```
MILAWA !>(%crewrite default)
; Rewrote clause #1 in 0.001 seconds (proved), [...]
; Rewrote 1 clauses; 0 (+ 0 forced) remain.
All goals have been proven.

MILAWA !>(%qed)
; Compiling worlds for NOT-IN-SELF...
...
;; Proof accepted.  Saving as user/thm-not-in-
self.proof
New rule: NOT-IN-SELF
```

The Milawa Interface

Unsound, Unverified

User Interface

=

Interfacing Nonsense

5,000 lines of ACL2 macros

Theorem Proving Tactics

2000 functions, 100,000 lines**

(Defined in the Logic)

- crewrite
- split
- use
- ...

This Talk

850K line design

Translator

125K loc

Model

350K loc

300K lisp lines

70K c/asm lines

A COMPUTATIONAL LOGIC

ACL2

APPLICATIVE COMMON LISP

+

clozure associates

GCC

2-4M lines

6-15M lines

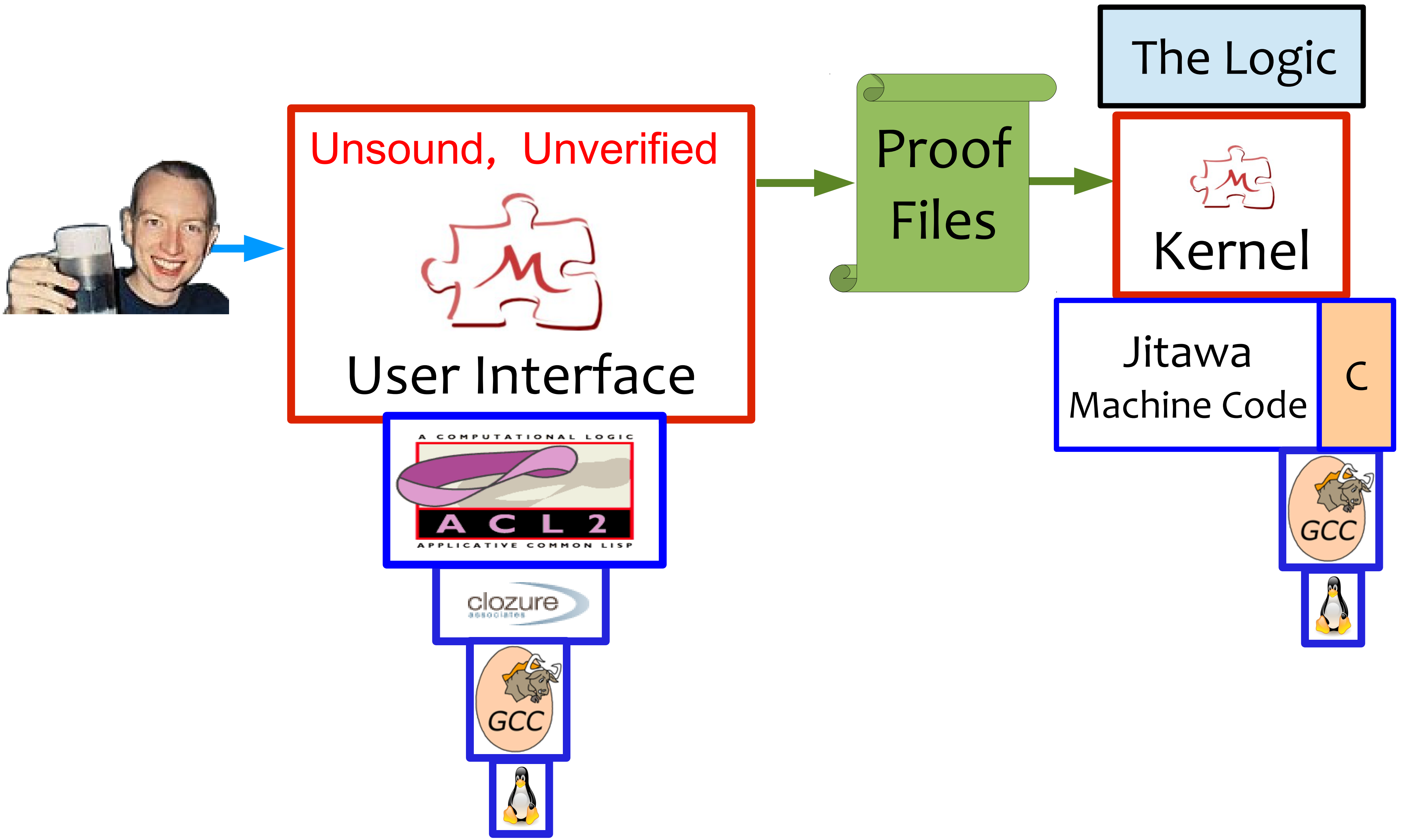
...ance

Human Effort

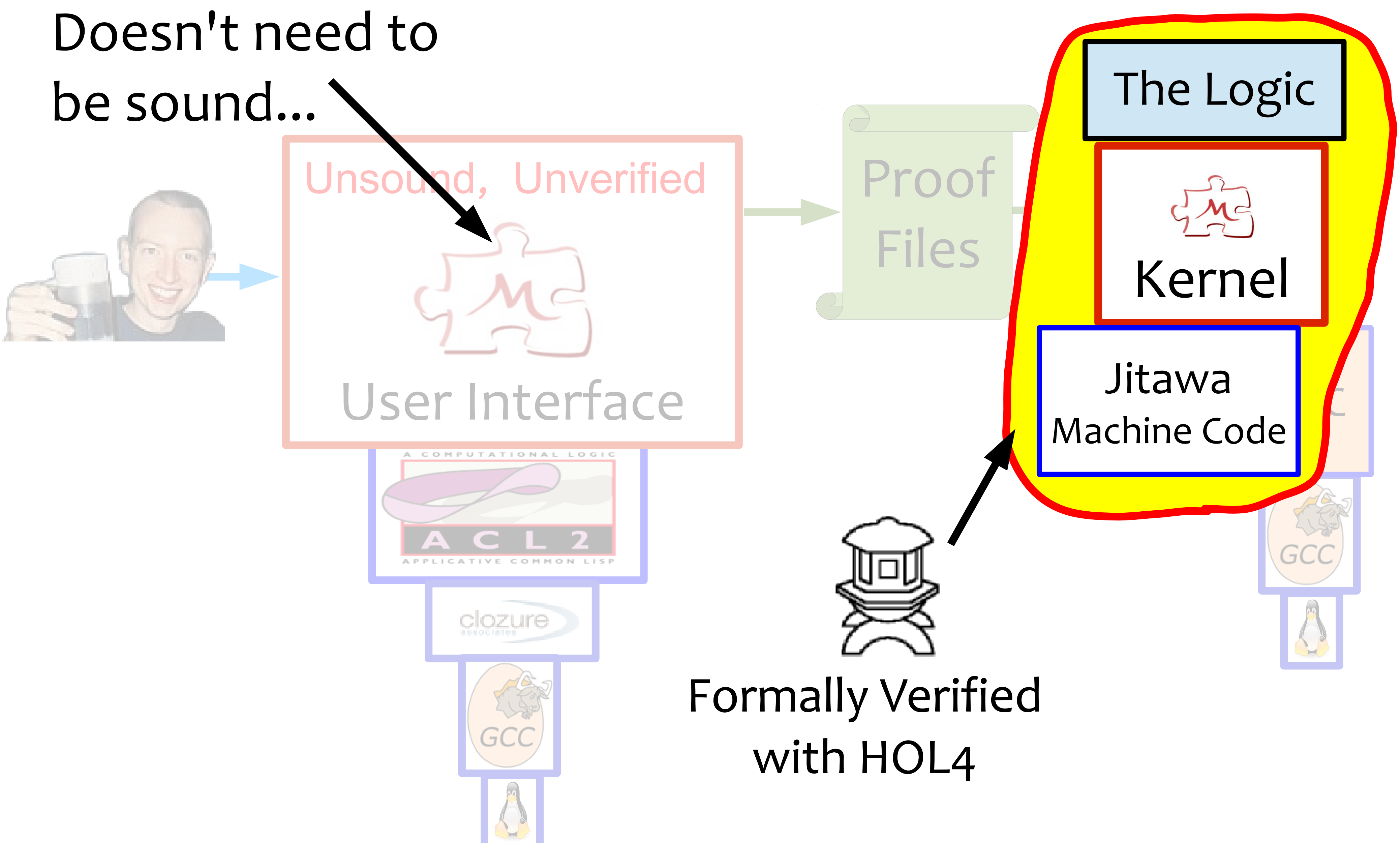
Properties

Heavy, heavy books

Milawa: A First Approximation



Foreshadowing



So what's in the Kernel?

Kernel =

Kernel (Lisp Program)
165 functions, 2000 lines incl. PC

Proof Checker
100 functions, 800 lines
(Defined in the Logic)

Define a function
Prove a theorem
Save your progress (checkpoint)
Switch to a new proof checker

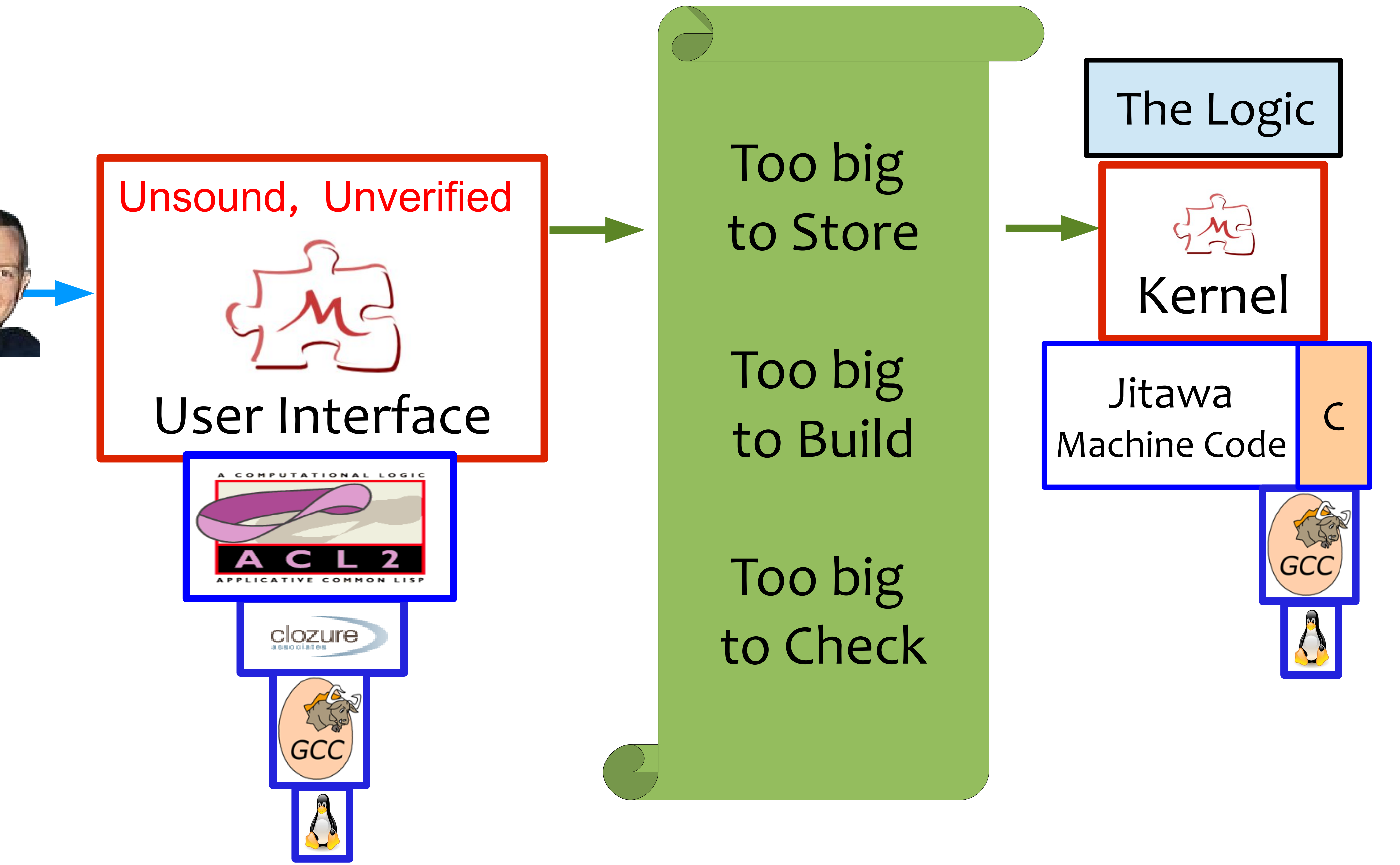
Teeny Tiny Proof Steps

But there's kind of a catch...

Teeny Tiny Proof Steps

Great Big Proof Files!

But there's kind of a catch...



Reflection and Self-Verification

Interfacing Nonsense

5,000 lines of ACL2 macros

Theorem Proving Tactics

2000 functions, 100,000 lines**

(Defined in the Logic)

crewrite

split

use

...

Kernel (Lisp Program)

165 functions, 2000 lines incl. PC

Proof Checker

100 functions, 800 lines

(Defined in the Logic)

Define a function

Prove a theorem

Save your progress (checkpoint)

Switch to a new proof checker

A Self-Verifying Theorem Prover

Interfacing Nonsense

Theorem Proving Tactics
2000 functions, 100,000 lines**
(Defined in the Logic)

Find, Writes

Bootstrapping Proofs
13,000 theorems, 8 GB on disk

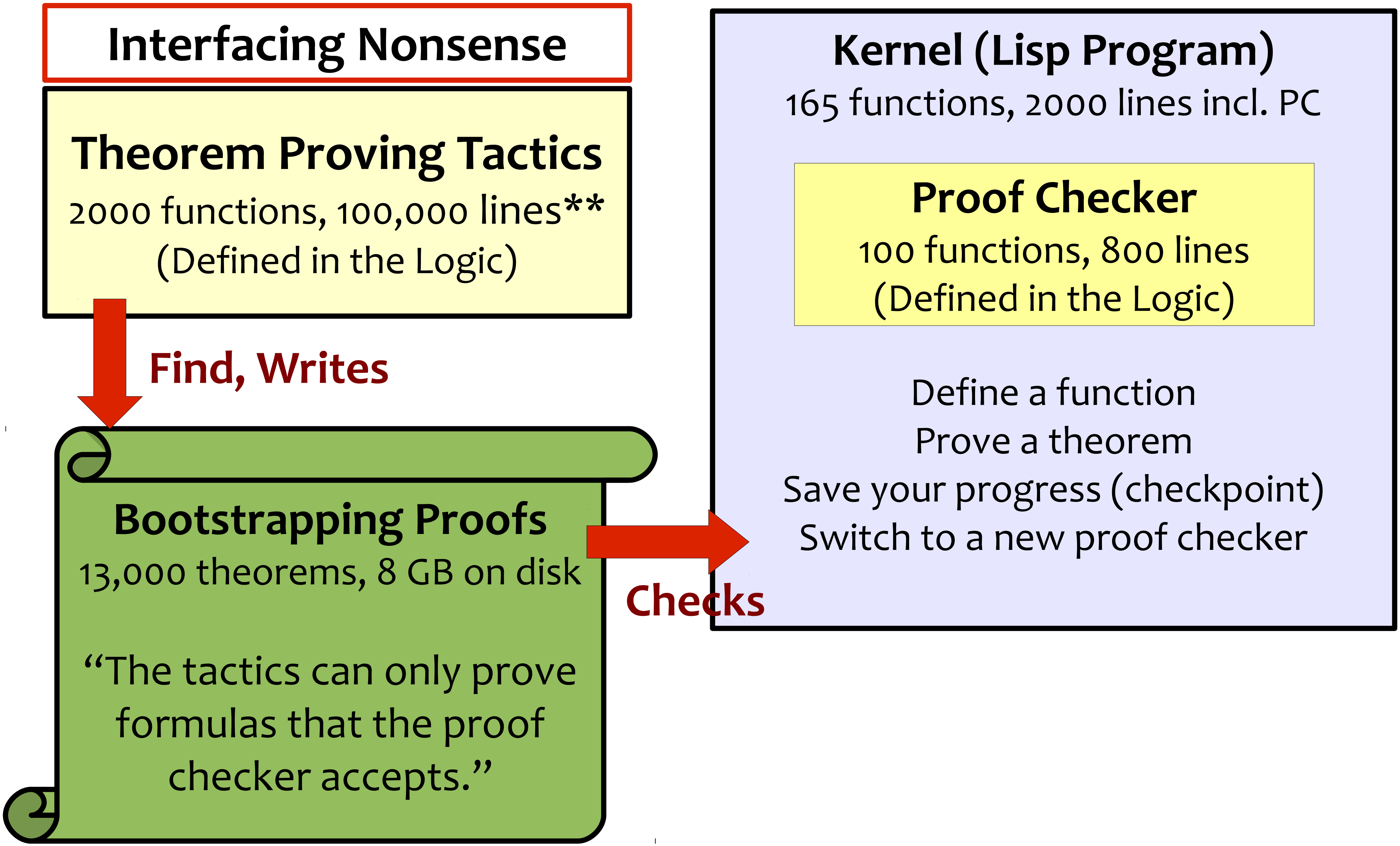
"The tactics can only prove formulas that the proof checker accepts."

Kernel (Lisp Program)
165 functions, 2000 lines incl. PC

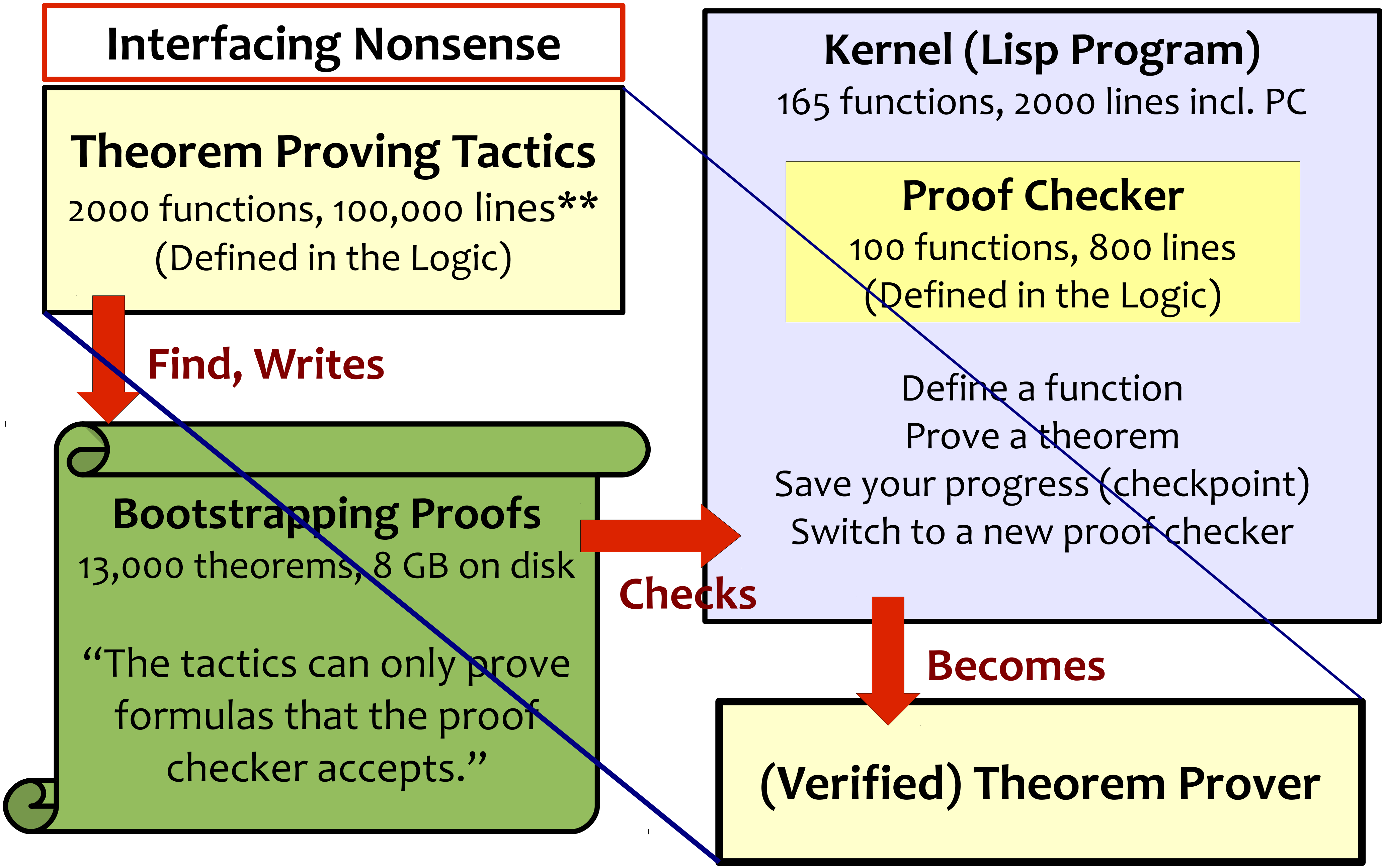
Proof Checker
100 functions, 800 lines
(Defined in the Logic)

Define a function
Prove a theorem
Save your progress (checkpoint)
Switch to a new proof checker

A Self-Verifying Theorem Prover



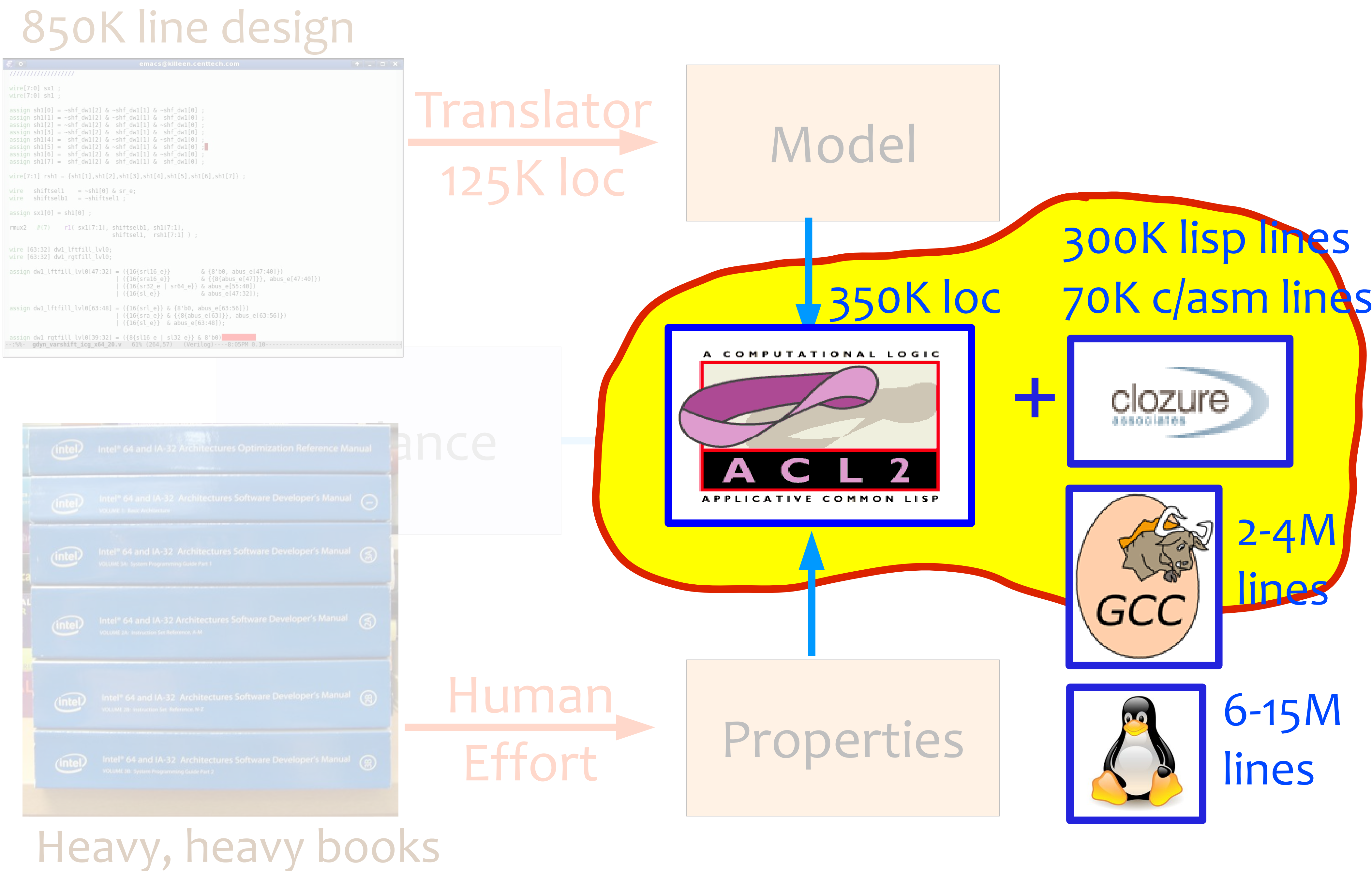
A Self-Verifying Theorem Prover



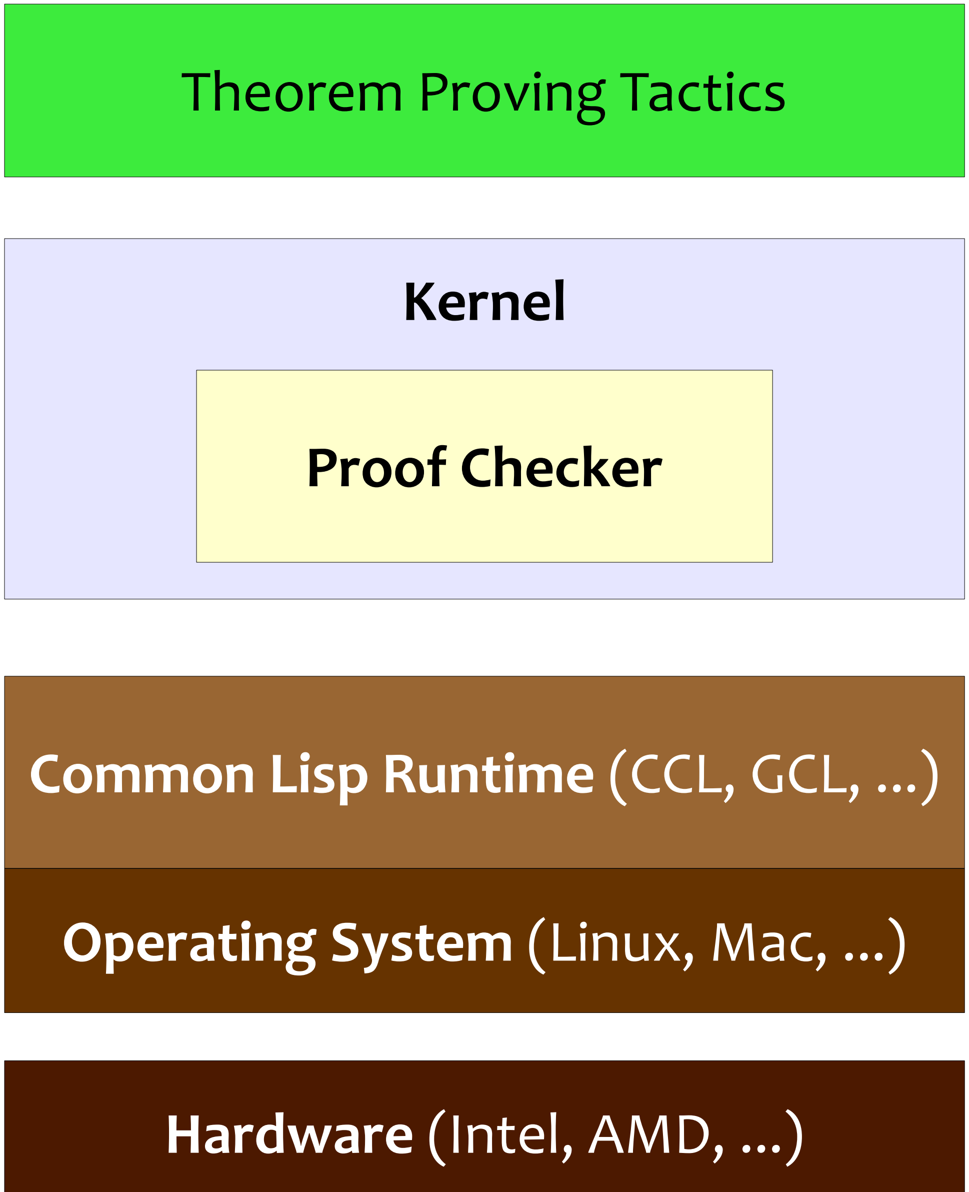
Bootstrapping

Level 11	All other tactics
Level 10	Conditional rewriter
Level 9	Evaluation, unconditional rewriting
Level 8	Rewrite traces
Level 7	Clause splitting
Level 6	Clause factoring, splitting groundwork
Level 5	Assumptions and clauses
Level 4	Miscellaneous groundwork
Level 3	Rules about basic functions
Level 2	Propositional reasoning
Core	Primitive rules of inference only

This Talk


850K line design
Translator
125K loc
Model
350K loc
300K lisp lines
70K c/asm lines
A COMPUTATIONAL LOGIC
ACL2
APPLICATIVE COMMON LISP
+
clozure associates
GCC
2-4M lines
6-15M lines
Human Effort
Properties
Intel® 64 and IA-32 Architectures Optimization Reference Manual
Intel® 64 and IA-32 Architectures Software Developer's Manual
Heavy, heavy books

The Soundness Story, So Far



Proven correct by the kernel

Is the logic sound?
Is the program faithful to it?

The program is short
"Social proof"

Practically have to trust
(no verified options)

Use multiple systems, at least

Fundamentally have to trust
Use multiple systems, at least

Two Projects Meet

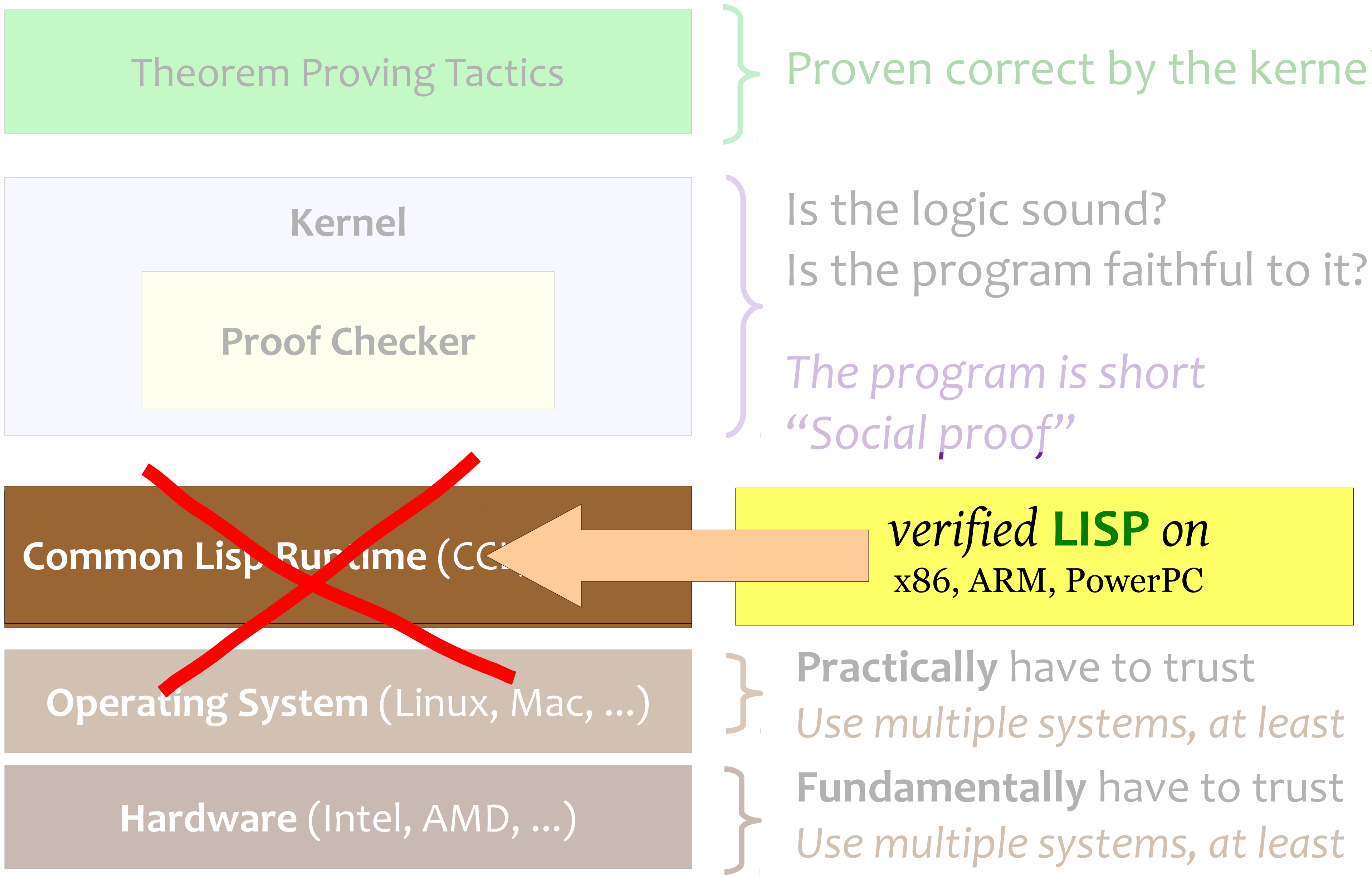
Milawa
Self-verifying theorem prover

Jared Davis, UT Austin, 2009

verified **LISP** *on*
x86, ARM, PowerPC

Magnus Myreen, Cambridge, 2008

So can we do *this*?



Well, no.

Bootstrapping Proofs
½ billion unique conses
16 hours on CCL
8 GB on disk

verified **LISP** *on*
x86, ARM, PowerPC

Interpreted, slow
32-bit, memory limited

Magnus set out to develop **Jitawa**, a new Lisp runtime for Milawa.

What does Milawa need?

Theorem Prover

First-order, recursive functions
Naturals, symbols, conses

12 Primitive Functions

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

11 Macros

```
and or list cond
let let*
first … fifth
```

Kernel

Destructive updates
Hash tables
File reading
Timing, status messages
Checkpointing
Function compilation
Dynamic function calls
Runtime errors

I/O Requirements

½ billion unique conses
8 GB on disk
Abbreviations are critical

What does Milawa *really* need?

Theorem Prover

First-order, recursive functions
Naturals, symbols, conses

12 Primitive Functions

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

11 Macros

```
and or list cond
let let*
first … fifth
```

Kernel

~~Destructive updates~~
~~Hash tables~~
~~File reading~~
~~Timing~~, status messages
~~Checkpointing~~
Function compilation
Dynamic function calls
Runtime errors

I/O Requirements

½ billion unique conses
~~8 GB on disk~~ 4 GB input file
Abbreviations are critical

Jitawa – A Scaled Up, Verified Lisp

Unverified C Wrapper

200 lines (with #if debug)

Parse command line

Allocate memory

Initialize IO function pointers

Invoke verified core

read_line

print_string

report_error

Verified Core

7500 lines of verified x86 machine code

Just-in-time (JIT) compiler to 64-bit x86

Copying garbage collector

Up to 2^{31} conses (16 GB), big stacks

Efficient parser with `#1=...` abbreviations

Always exits gracefully

Calls C routines for I/O

Far simpler than a full Common Lisp implementation

Jitawa – A Scaled Up, Verified Lisp

Unve

200 li

Pa

A

Initializ

In

report_error

```
> '3
3
> (cons '5 '(6 7))
(5 6 7)
> (define 'increment '(n) '(+ n '1))
NIL
> (increment '5)
6
```

code

86

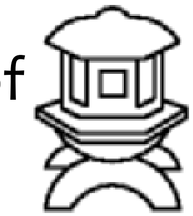
eviations

Implements an ordinary read-eval-print loop!

How is it Verified?

Jitawa Specification
400 lines of HOL
Parsing, Evaluation, Printing

HOL Proof
Effort



Compiler, GC, Parsing, Printing
Algorithms Defined in HOL

X86-64 Model
(in HOL)

Not a full x86 model
Just the relevant instructions

HOL Proof

Proof producing synthesis

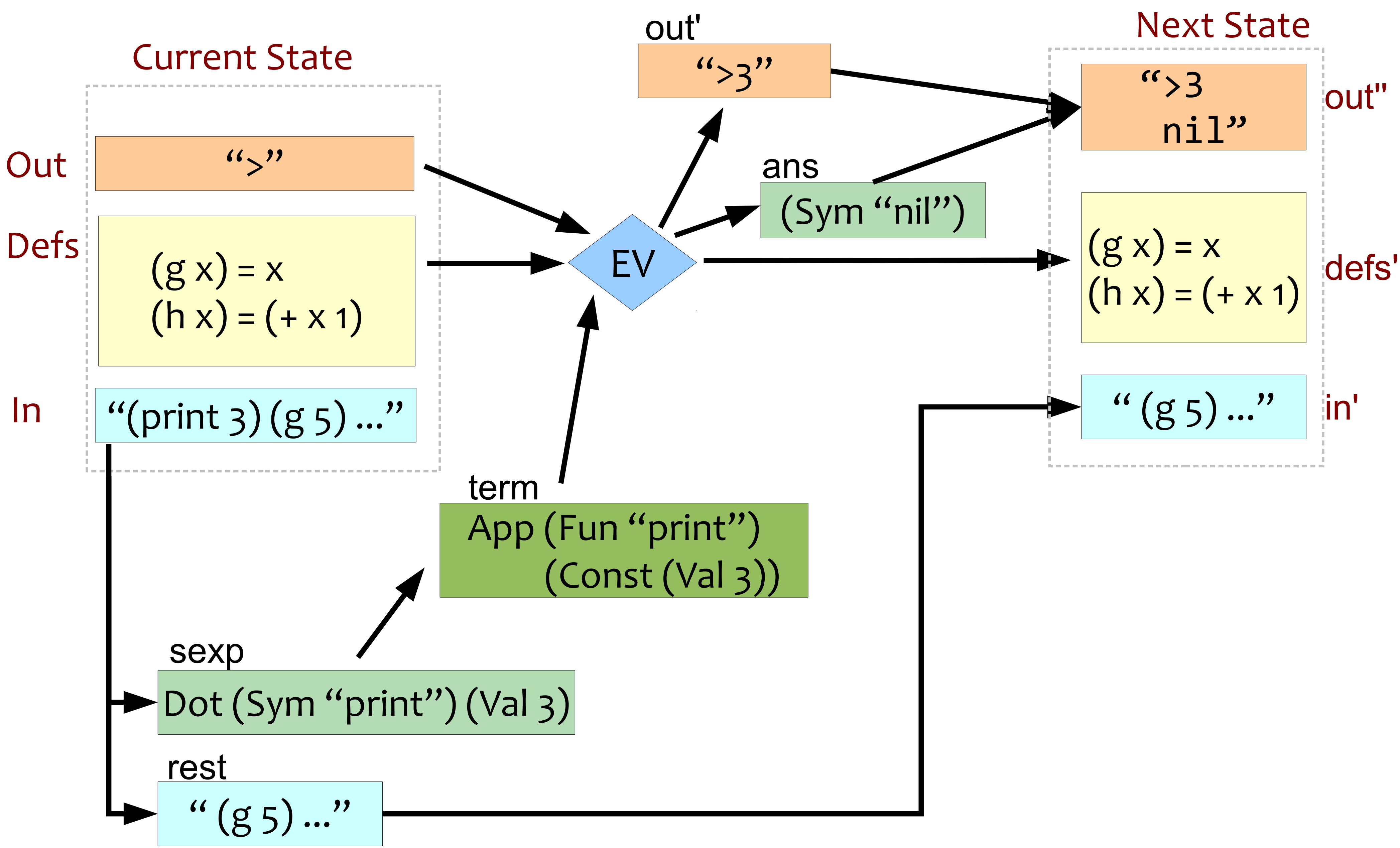
Jitawa Implementation
X86-64 Machine Code

flikr:vext01

\+ Testing

wikipedia

Jitawa Specification (400 lines of HOL)



An Improved Soundness Story

Theorem Proving Tactics	Proven correct by the kernel
Kernel / **Proof Checker**	Is the logic sound? Is the program faithful to it? *The program is short* *“Social proof”*
Jitawa Verified Core / C	Proven correct down to the Machine Code in HOL4
Operating System (Linux, Mac, …)	**Practically** have to trust *Use multiple systems, at least*
Hardware (Intel, AMD, …)	**Fundamentally** have to trust *Use multiple systems, at least*

Lifting

Simplified Kernel

1700 Lines of Jitawa Lisp

Jitawa Specification

400 lines of HOL

Parsing, Evaluation, Printing

Lifting

Simplified Kernel

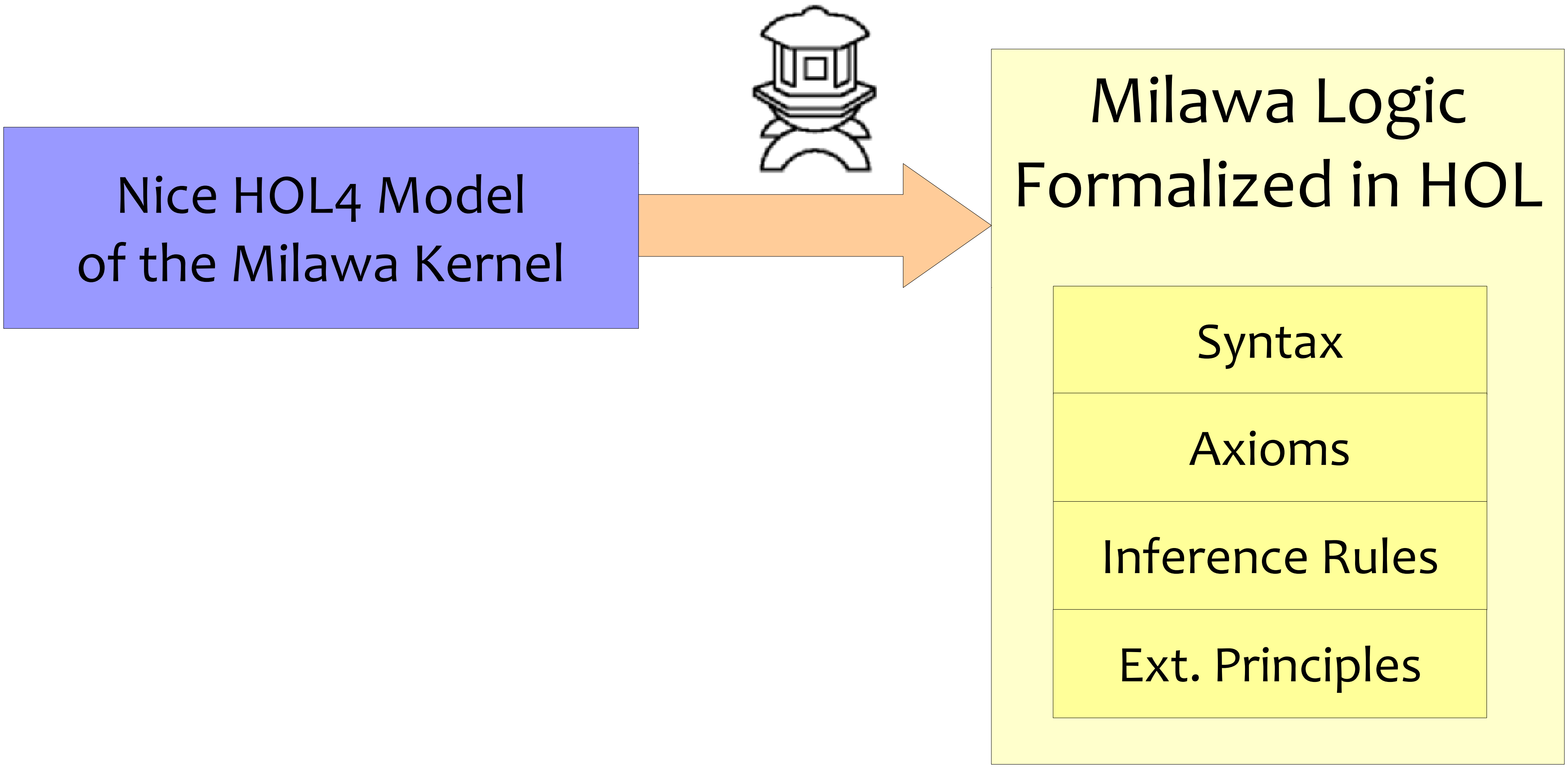
1700 Lines of
Jitawa Lisp

Jitawa Specification
400 lines of HOL
Parsing, Evaluation, Printing

Automated
(HOL4 tactics)

Nice HOL4 Model
of the Milawa Kernel

Faithfulness



Soundness

Milawa Logic Formalized in HOL

- Syntax
- Axioms
- Inference Rules
- Ext. Principles

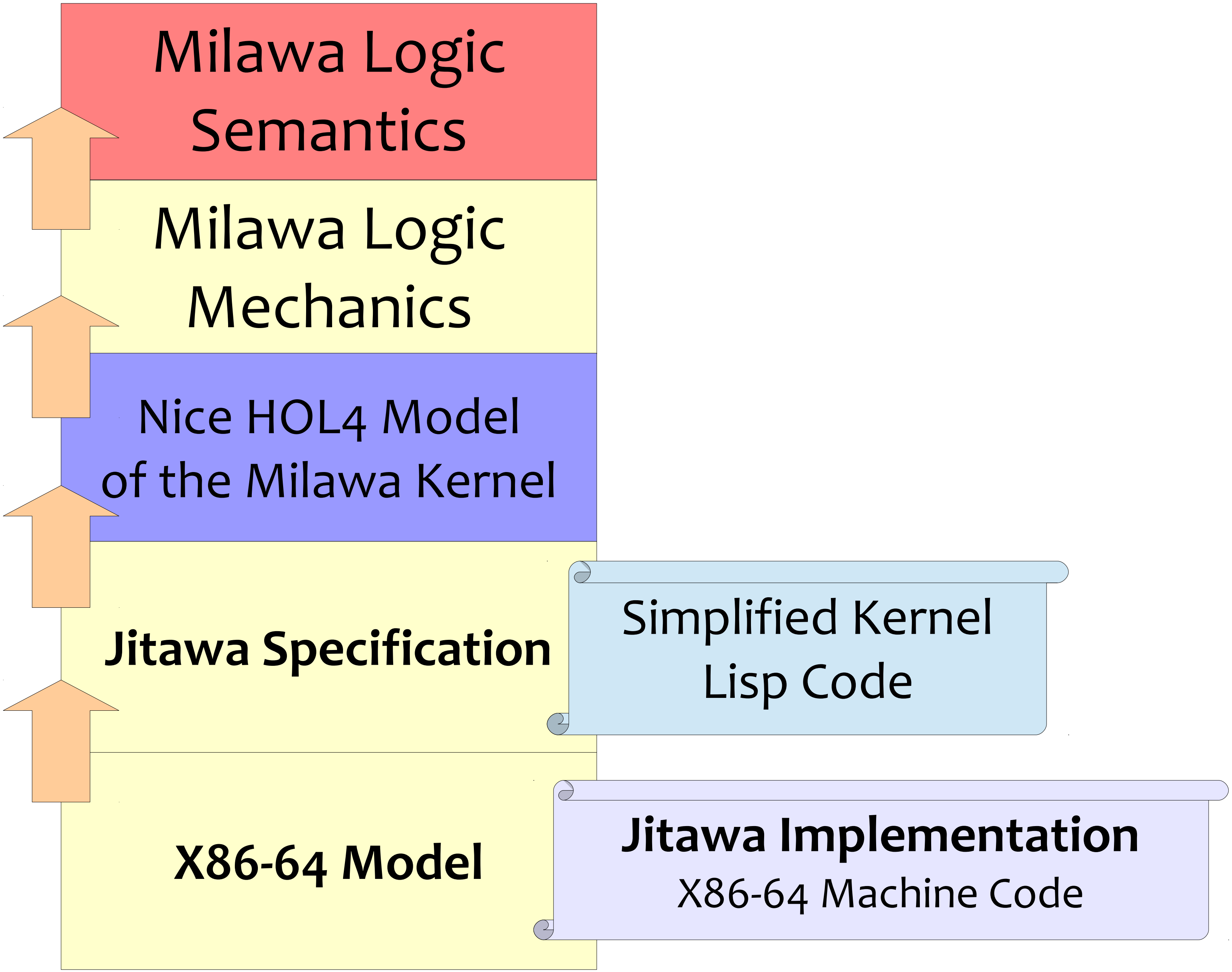
→ Milawa Logic Semantics

Syntactically provable

$$\forall \pi\ p.\ \ \text{context\_ok}\ \pi \wedge (\vdash_\pi p) \Longrightarrow (\models_\pi p)$$

Semantically true

Putting it all Together



Putting it all Together

Milawa Logic Semantics

$$
\begin{array}{l}
\forall \mathit{input}\ p.\\
\quad \{\,\mathsf{init\_state}\ (\mathsf{milawa\_implementation} \mathbin{+\!\!+} \texttt{"(milawa-main '}\mathit{input}\texttt{)"}) * \mathsf{pc}\ \mathit{pc}\,\}\\
\quad\ \mathit{pc} : \mathsf{code\_for\_entire\_jitawa\_implementation}\\
\quad \{\,\mathsf{error\_message} \lor \exists \mathit{output}.\ \langle \mathsf{every\_line}\ \mathsf{line\_ok}\ \mathit{output} \rangle\ *\\
\qquad\qquad\qquad\qquad \mathsf{final\_state}\ (\mathit{output} \mathbin{+\!\!+} \texttt{"SUCCESS"})\,\}
\end{array}
$$

Jitawa Specification

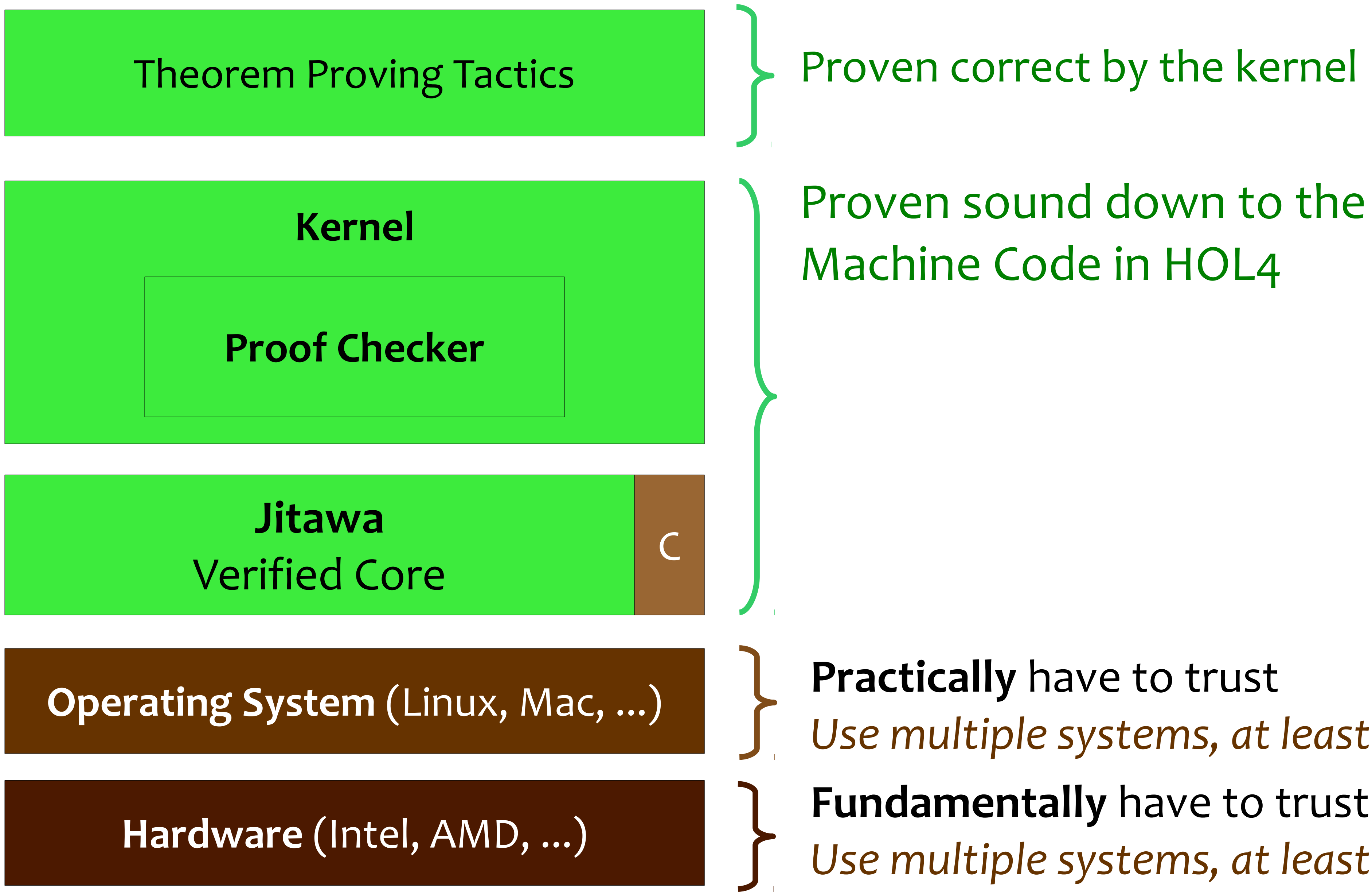
Lisp Code

X86-64 Model

Jitawa Implementation

X86-64 Machine Code

The New Soundness Story



Thanks!

Questions?