A Verified Runtime for a Verified Theorem Prover

Magnus Myreen
University of Cambridge, UK

Jared Davis
Centaur Technology, USA

This work is published in ITP '11
Slides for ACL2 '11

Two Projects Meet

Milawa

Self-verifying theorem prover

Jared Davis, UT Austin, 2009

A Self-Verifying Theorem Prover

Theorem Prover

2000 functions, 100,000 lines**

(Defined in the Logic)

Proof Checker

100 functions, 800 lines

(Defined in the Logic)

A Self-Verifying Theorem Prover

Theorem Prover

2000 functions, 100,000 lines**

(Defined in the Logic)

Finds, Writes ("ahead of time")

Bootstrapping Proofs

13,000 theorems, 8 GB on disk

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

Proof Checker

100 functions, 800 lines

(Defined in the Logic)

A Self-Verifying Theorem Prover

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

Finds, Writes ("ahead of time")

Bootstrapping Proofs
13,000 theorems, 8 GB on disk

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

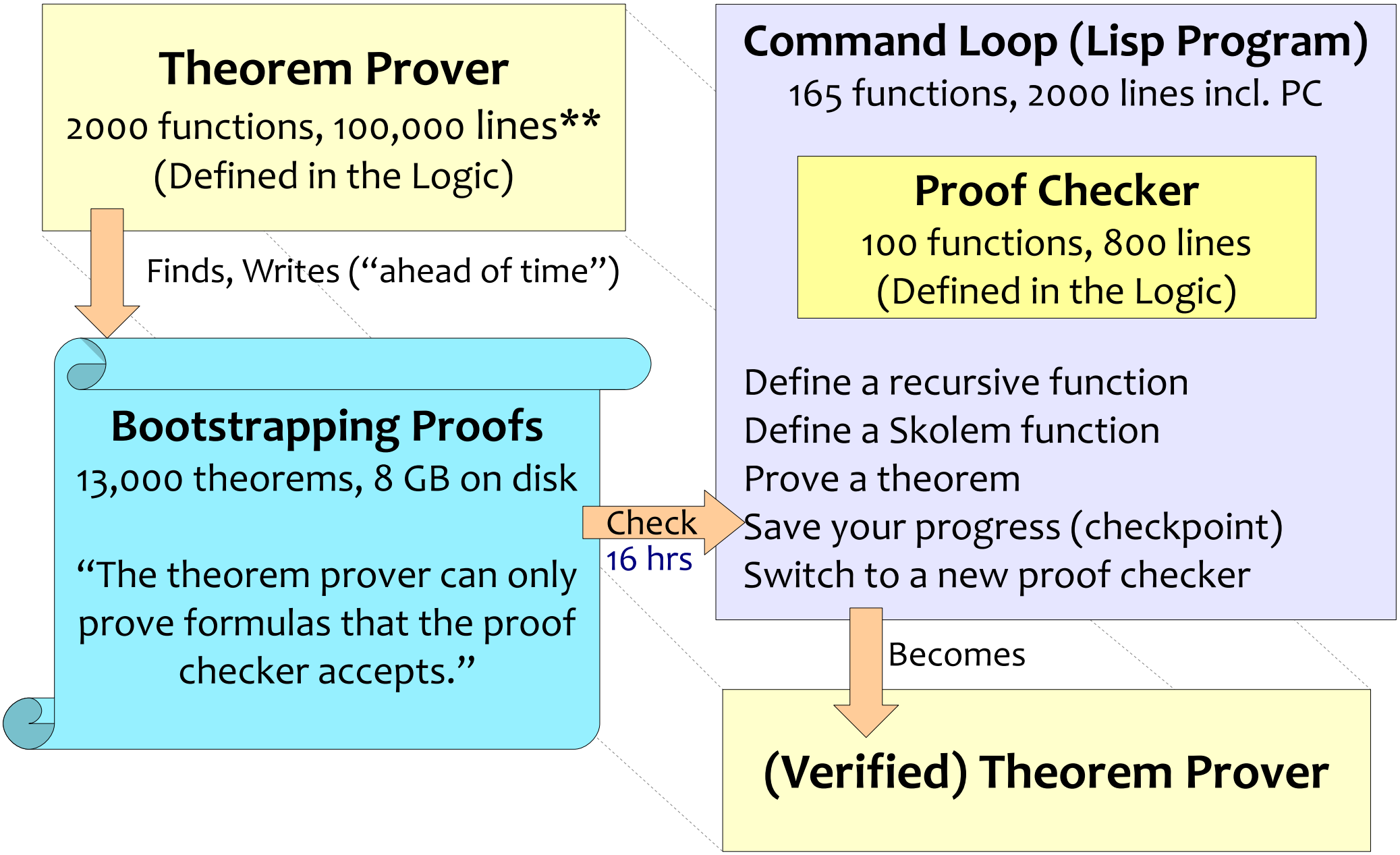
Check
16 hrs

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

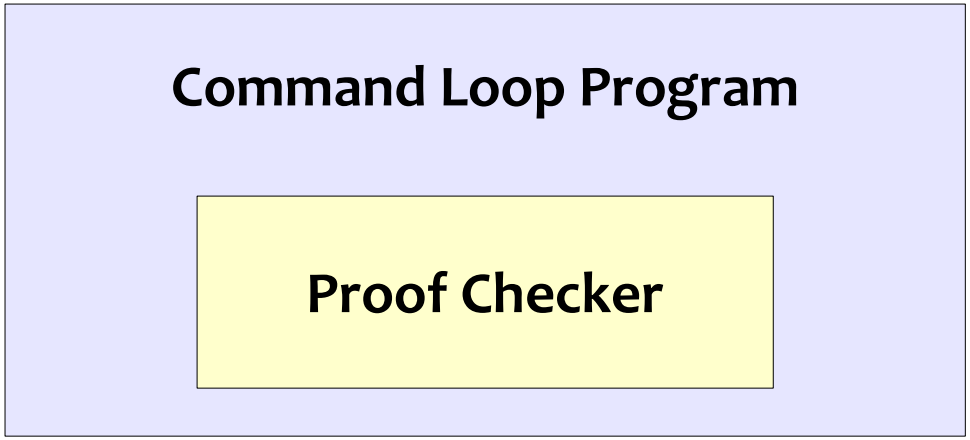
Proof Checker
100 functions, 800 lines
(Defined in the Logic)

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

A Self-Verifying Theorem Prover

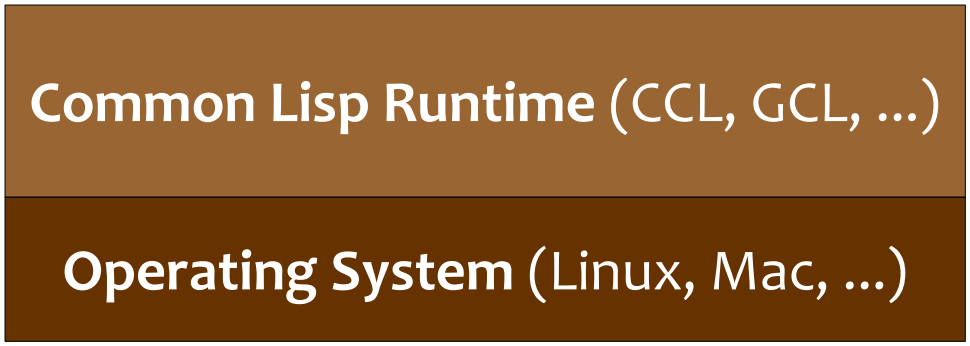


The Soundness Story



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

The program is short
Social proof, for now



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

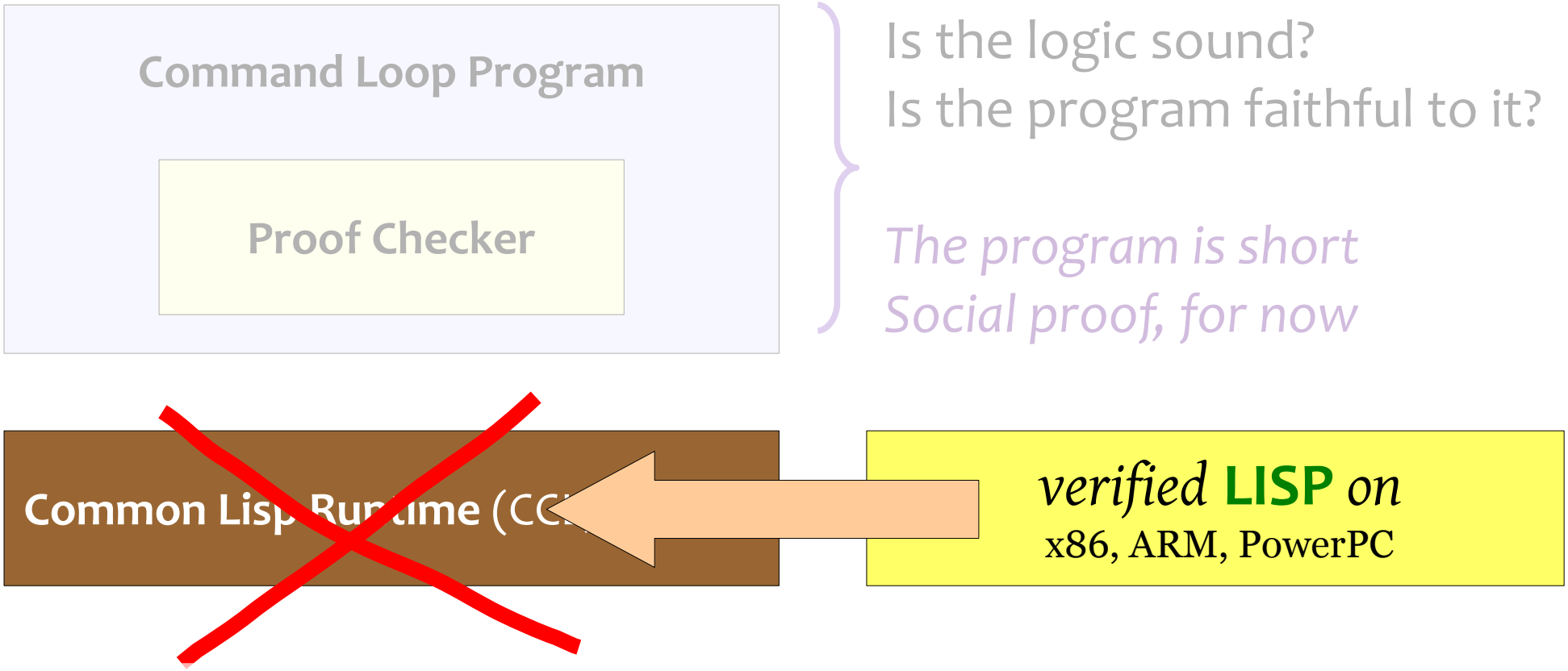
Jared Davis, UT Austin, 2009

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
```

Command Loop

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
```

Command Loop

~~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 Scalable, 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

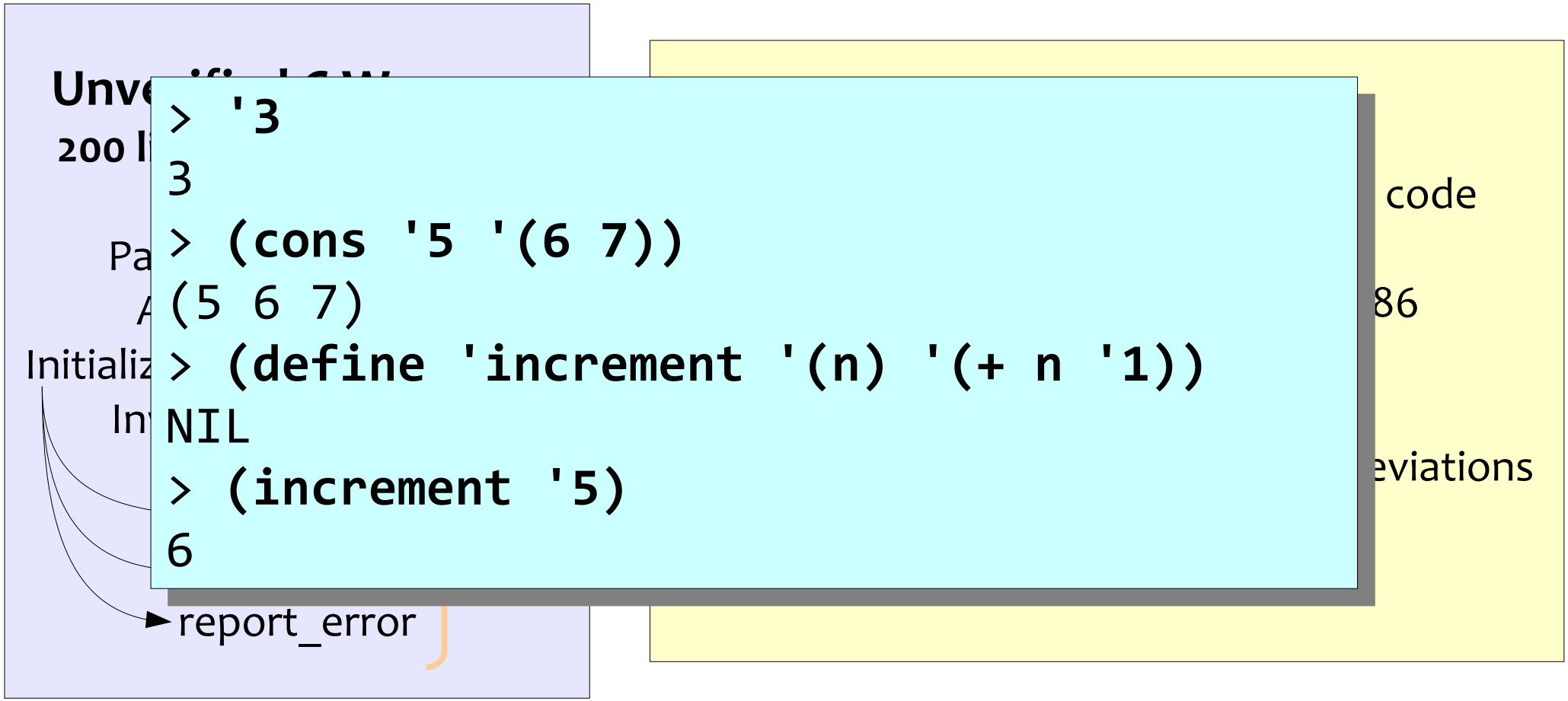
Verified Core

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= style abbreviations
Graceful exit in all circumstances
Calls C routines for I/O

Compare trusting this to an ordinary Lisp implementation

Jitawa – A Scalable, Verified Lisp



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

Machine Code

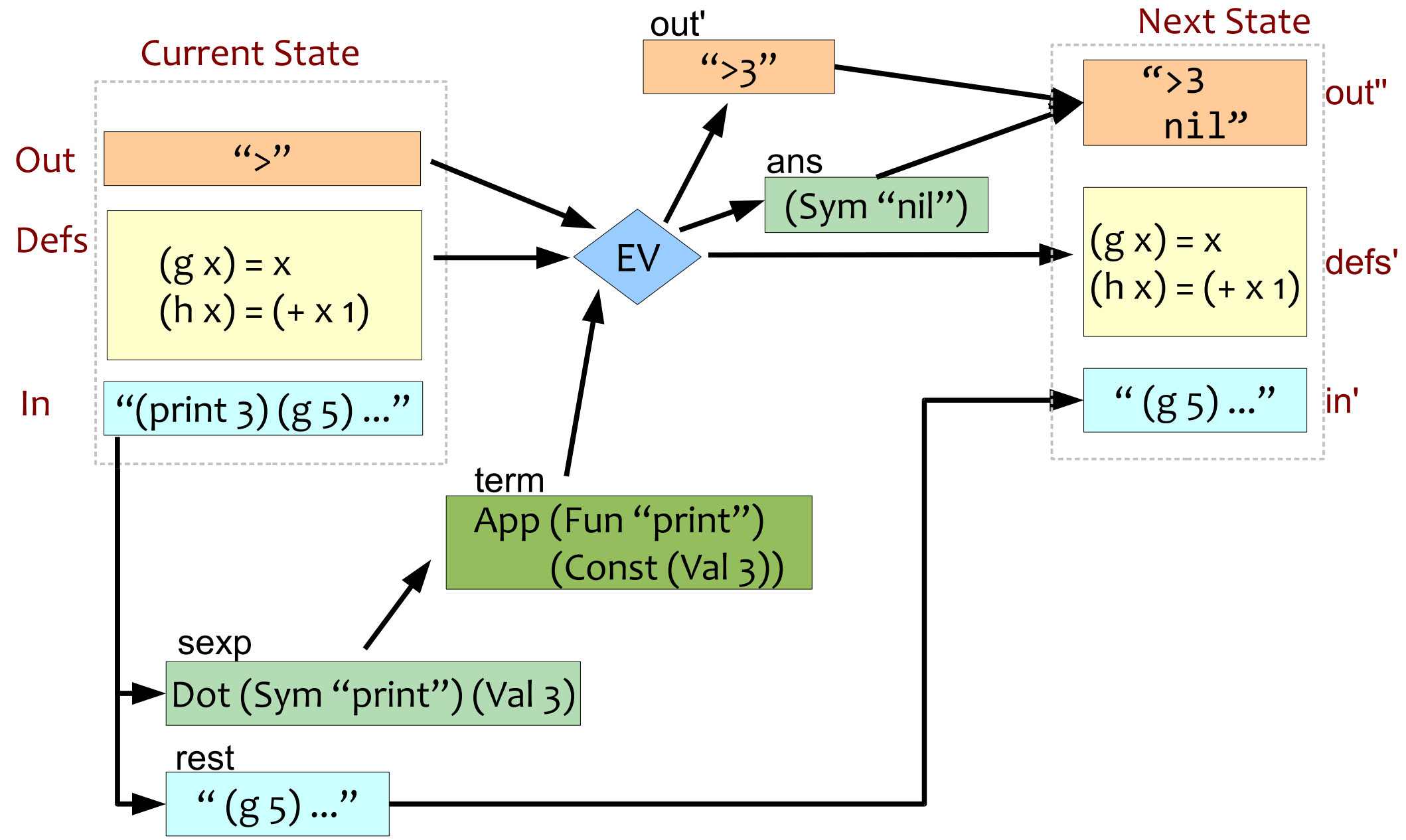
flikr:vext01

\+ Testing

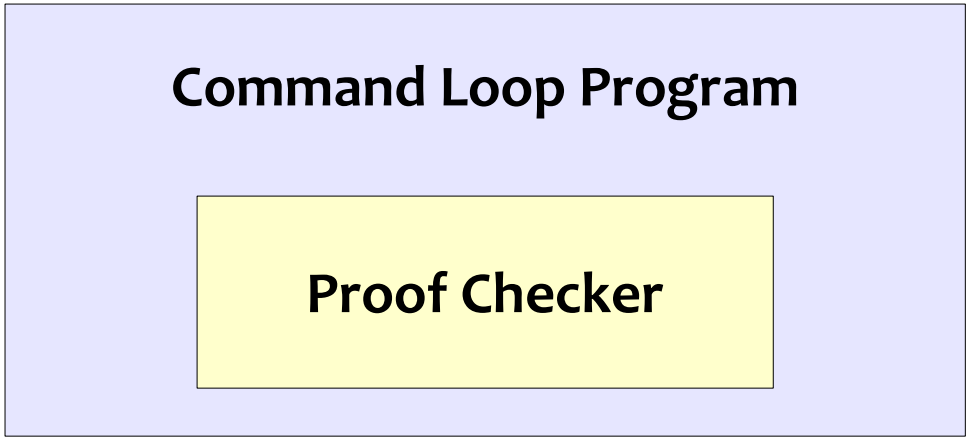
wikipedia

Magnus Myreen

Jitawa Specification (400 lines of HOL)



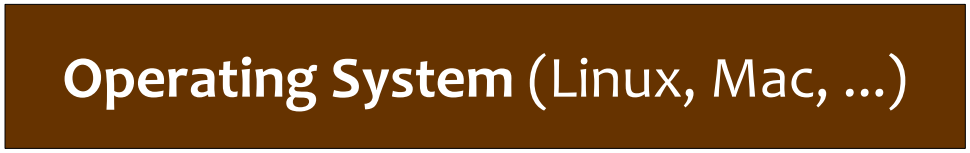
The New Soundness Story



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

The program is short
Social proof, for now

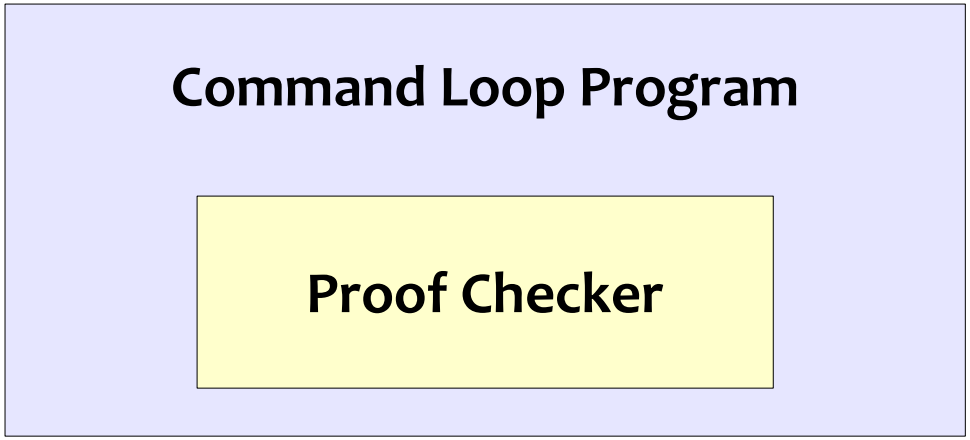
Verified Down to the
Machine Code in HOL4



Practically have to trust
(no verified options)
Use multiple systems, at least

Fundamentally have to trust
Use multiple systems, at least

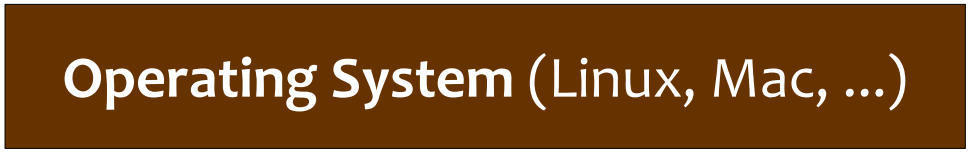
Future Work



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

Mechanize this in HOL4,
Connect it to the Runtime!

Verified Down to the
Machine Code in HOL4



Practically have to trust
(no verified options)
Use multiple systems, at least

Fundamentally have to trust
Use multiple systems, at least