Steps Towards Verified Implementations of HOL Light

Magnus O. Myreen[1], Scott Owens[2], Ramana Kumar[1]

[1] University of Cambridge, UK
[2] University of Kent, UK

ITP 2013

Background

Could your verified Lisp run my ACL2-like prover?

Jared Davis (UT Austin)

Milawa

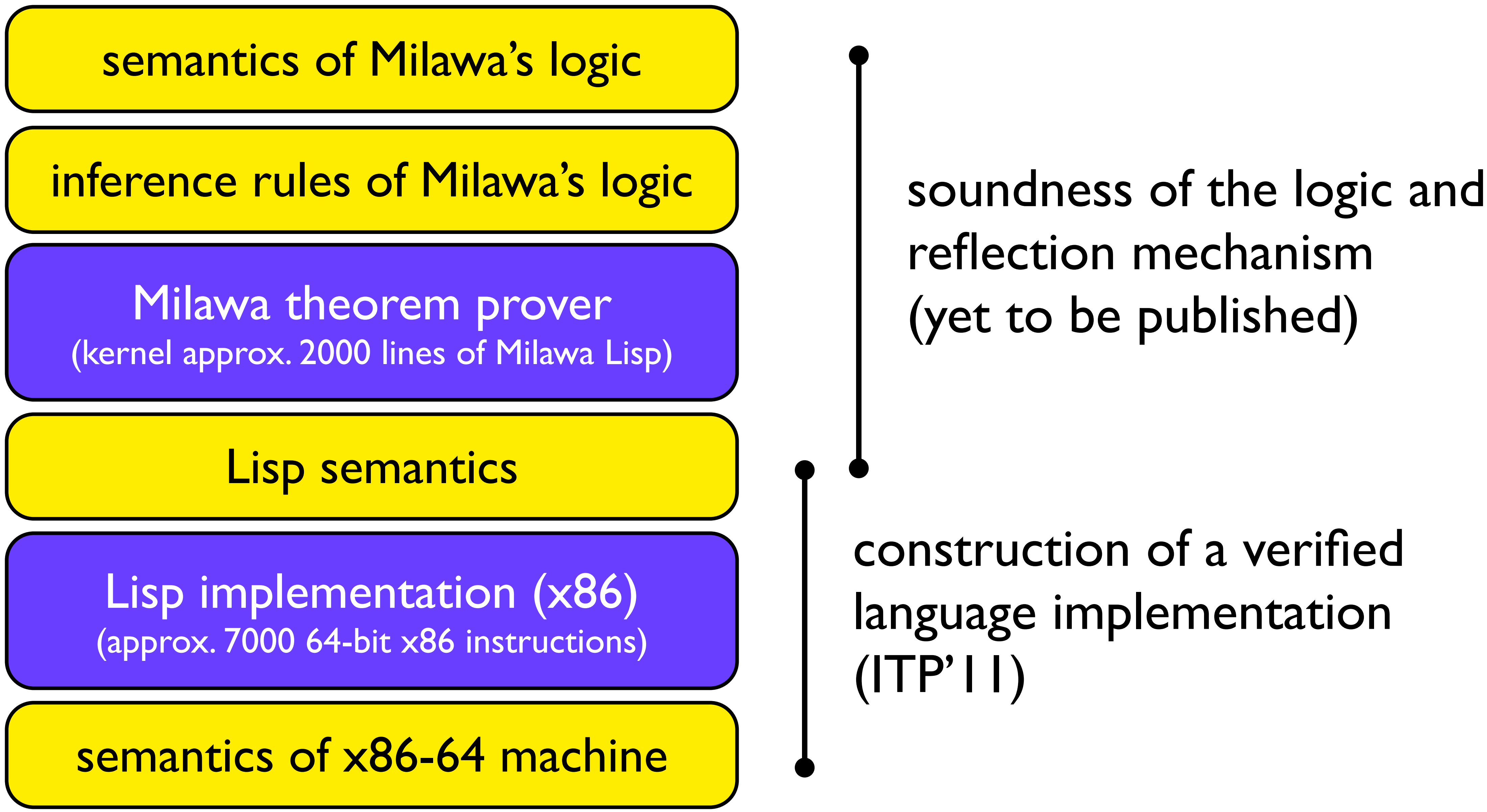
"a self-verifying theorem prover"
(Davis' PhD thesis)

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

Jitawa: verified **LISP** *with JIT compiler*

(TPHOLs 2009)

Proving Milawa sound



At ITP'11

Please, do the same for HOL light!

Freek Wiedijk
Radboud University Nijmegen

My immediate response:

That would be difficult...

Later thought:

Maybe...

A new project:

CakeML

A verified implementation of ML

"The CakeML language is designed to be both easy to program in and easy to reason about formally"

People involved

Ramana Kumar
(Uni. Cambridge)

operational **semantics**

verified **compilation** from CakeML to bytecode

Scott Owens
(Uni. Kent)

verified **type** inference

Michael Norrish
(NICTA, ANU)

verified **parsing** (syntax is compatible with SML)

verified **x86** implementations

proof-producing **code generation** from HOL

Magnus Myreen
(Uni. Cambridge)

CakeML version of

Soundness of HOL light

semantics of HOL with defns

inferences of HOL with defns

HOL light kernel in CakeML
(module consisting of ~500 lines of CakeML)

CakeML op. semantics

CakeML implementation
(a read-eval-print loop in 64-bit x86 code)

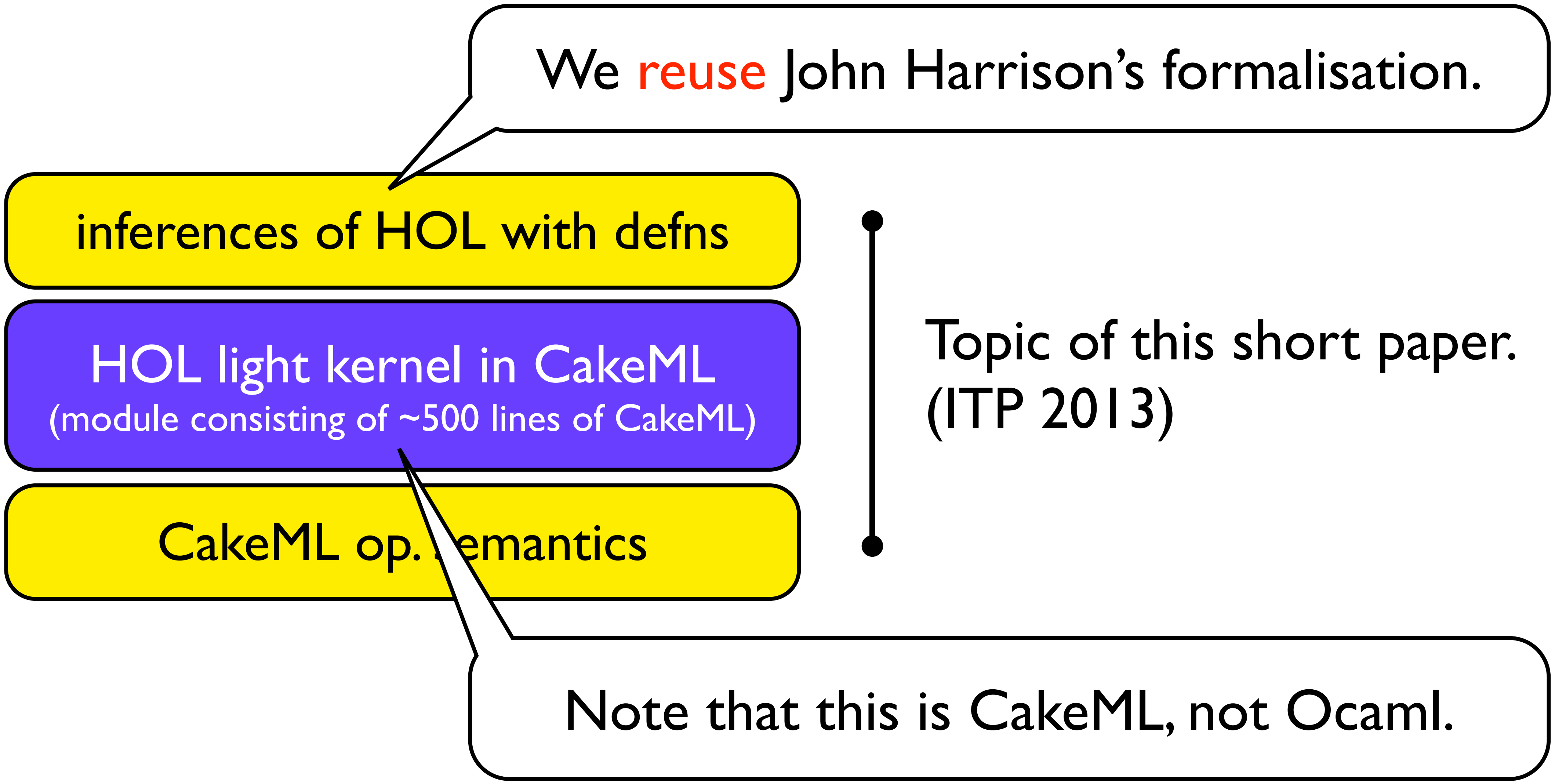
semantics of x86-64 machine

John Harrison's proof (IJCAR 2006) but *without definitions*

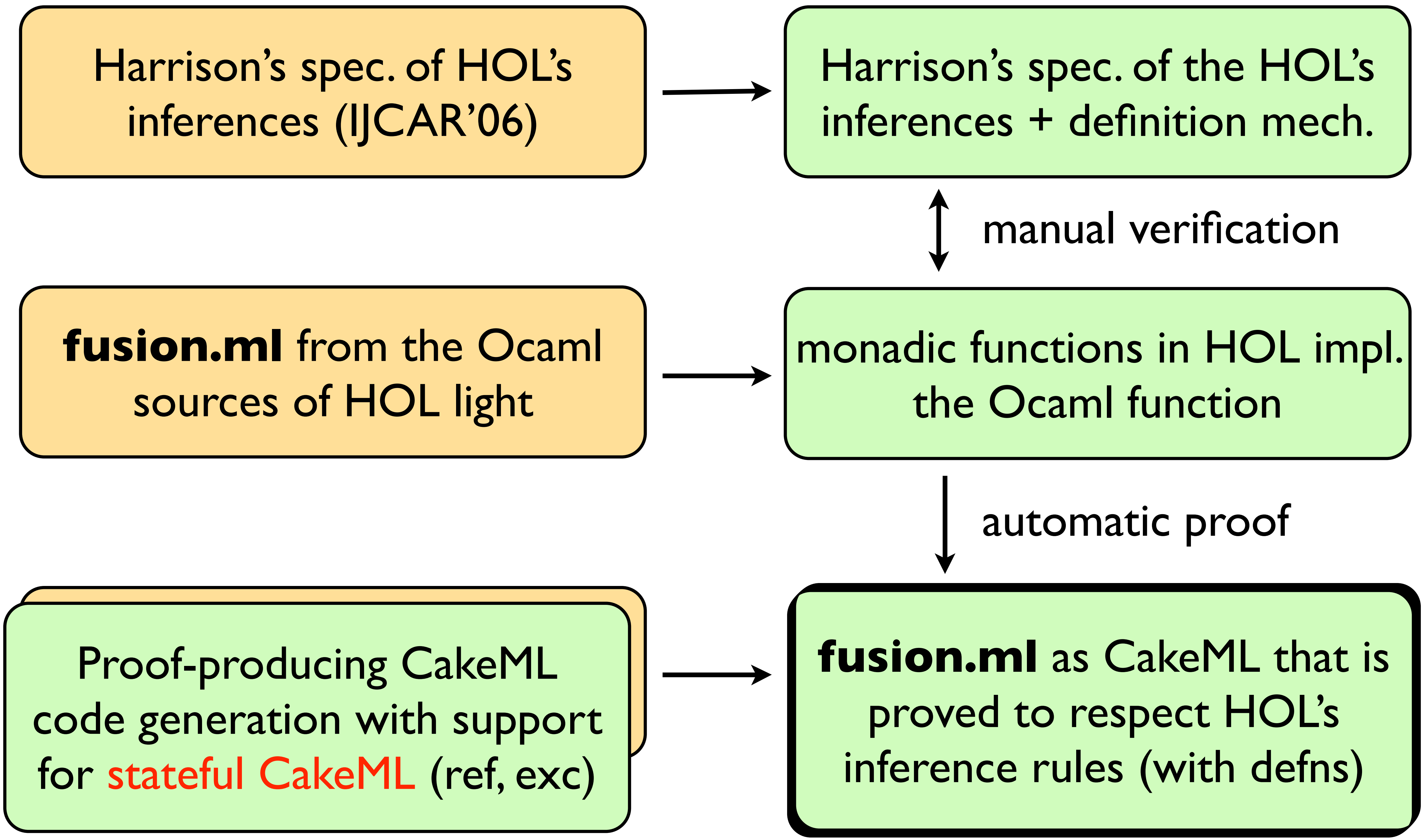
Topic of this short paper. (ITP 2013)

nearly complete, ask for details!

This talk


We reuse John Harrison's formalisation.
inferences of HOL with defns
HOL light kernel in CakeML
(module consisting of ~500 lines of CakeML)
CakeML op. semantics
Topic of this short paper.
(ITP 2013)
Note that this is CakeML, not Ocaml.

Approach



Harrison's spec. of HOL's inferences (IJCAR'06)
Harrison's spec. of the HOL's inferences + definition mech.
manual verification
fusion.ml from the Ocaml sources of HOL light
monadic functions in HOL impl. the Ocaml function
automatic proof
Proof-producing CakeML code generation with support for stateful CakeML (ref, exc)
fusion.ml as CakeML that is proved to respect HOL's inference rules (with defns)

More concretely

For each Ocaml function in **fusion.ml**,

```
let REFL tm = Sequent([],mk_eq(tm,tm))
```

we define a monadic function in HOL:

```
REFL tm = do eq <- mk_eq(tm,tm);
             return (Sequent [] eq) od
```

prove that this shallow embedding respects the inferences and use proof-producing code generation to produce CakeML:

```
val REFL = fn tm =>
  let val eq = mk_eq (tm, tm)
  in Sequent ([], eq) end;
```

Summary

Main message of the talk:

We are working towards a verified implementation of ML (called CakeML)

A verified HOL light is an initial challenge case study for CakeML.

Current status of the project on next slide...

Current status

semantics of HOL with defns

inferences of HOL with defns

HOL light kernel in CakeML
(module consisting of ~500 lines of CakeML)

CakeML op. semantics

CakeML implementation
(a read-eval-print loop in 64-bit x86 code)

semantics of x86-64 machine

John Harrison's proof (IJCAR 2006) but *without definitions*

Topic of this short paper.
(ITP 2013)

Lift soundness of kernel to soundness result for entire prover

nearly complete, ask for details!