Defining a Niche for HOL4

Michael Norrish

August 2015

Outline

1. State of Play
2. Why Keep Going?
3. If We Do, Where Do We Go?

Context

The world has many interactive theorem-proving systems.

Coq and Isabelle (at least) have larger user bases.

Big systems get more developer love:

- HOL4 needs to “choose its battles”

Strengths (Inherent)

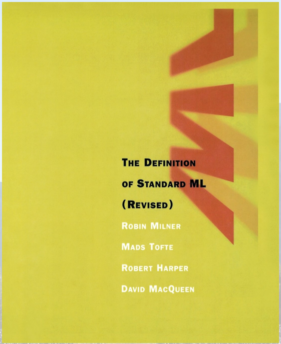
- SML
- HOL
- Persistence Model
- Tools à la Unix?
- ?

[Lake Kananaskis by `davebloggs007@hotmail.com` *via* Flickr]

SML as a Strength

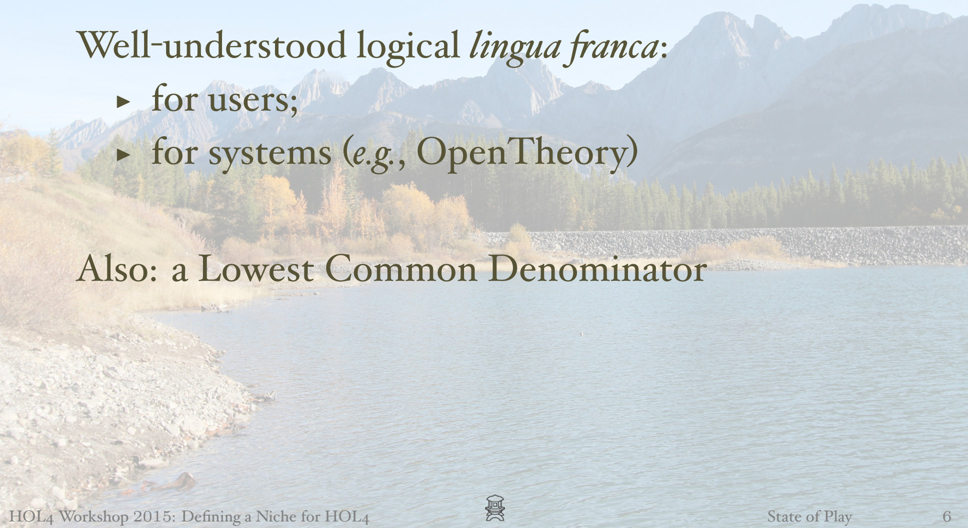
Well-defined language.

Clean semantics.



Has the features the implementor wants:

- type system
- exceptions
- even concurrency (in Poly/ML)



HOL as a Strength

Well-understood logical *lingua franca*:

- for users;
- for systems (*e.g.*, OpenTheory)

Also: a Lowest Common Denominator

Persistence Model

Theories are available on disk in an **implementation-independent** way.

MoscowML and Poly/ML implementations use the same format.

Strengths (Accidental/Historical)

Existing Formalisations

- CakeML, hardware models, ...

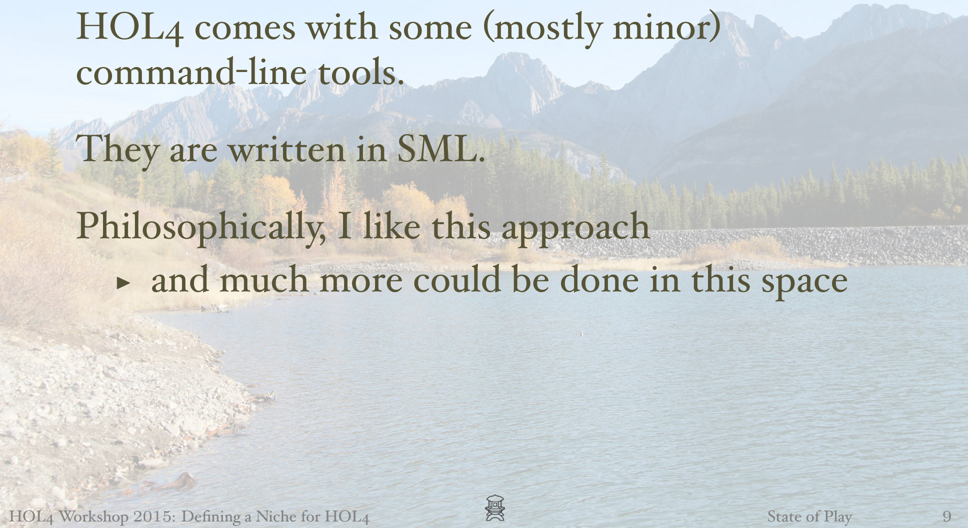
Existing Users

Documentation

Minimal code churn

- caused by slow development...

Unix-style Tools



HOL4 comes with some (mostly minor) command-line tools.

They are written in SML.

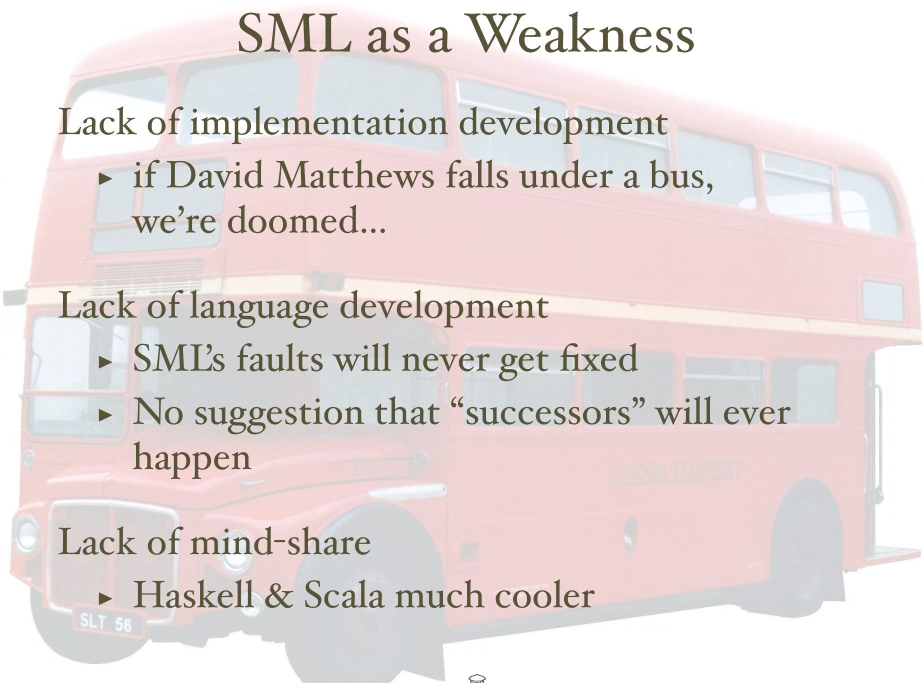
Philosophically, I like this approach

- and much more could be done in this space

Weaknesses

- SML
- HOL
- Windows
- User Interface
- Persistent Theories as Code
- Script-files as Code
- Lack of Concurrency

SML as a Weakness



Lack of implementation development

- if David Matthews falls under a bus, we're doomed...

Lack of language development

- SML's faults will never get fixed
- No suggestion that "successors" will ever happen

Lack of mind-share

- Haskell & Scala much cooler

HOL as a Weakness

HOL doesn't have cool types.

Not even Isabelle/HOL's type-classes.

- And lacking constants with different definitions on different types fundamentally blocks some constructions

User Interface

The `emacs` mode is hobbled by script files as code.

- Some would swear by `emacs` as an IDE
- ...but probably not for SML

Maybe proofs need different editing tools compared to code.

Too Much Code; Not Enough Data

Script files as SML code—yuck.

Theory files as SML code—yuck.

- We were too taken with the idea of getting namespace management from the language implementation
- Script files as data would do away with the need for this "advantage"

HOL on Windows

A sub-par experience:

- Moscow ML is slow
- Without `emacs`, users don't get Unicode

(lack of external dependencies is nice though...)

Why Keep Going?

Selfishness

HOL4 is “owned” by a relatively small group of people.

It is (relatively) easy to push it around according to that group’s taste.

- It’s not even that hard to become an “owner”

So: why give up on a system that can be what *I* want it to be?

One Riposte

Maybe *I* want a system with

- a great UI;
- powerful use of concurrency;
- declarative proof; and
- cutting edge logical tools
  - *e.g.*, powerful datatypes, code evaluation...

Are *you* going to provide all that?

A Scary Alternative

Would the world be better off if:

- we ported all HOL4 work to Isabelle/HOL?

Theorems probably wouldn't be hard to port.

- Large models/definitions may already exist in prover-independent form

Tools would be more of a challenge, but clearly possible in principle.

Dismissing Scary Alternatives

No-one is standing up to do all that work.

HOL4's existing users are probably mostly happy with it as is.

So

Let's Do Nothing (?)

But We Want a System With a Future
(I suppose)

Can HOL4 remain the preserve of a

- small,
- barely self-perpetuating

group of users?

It's harder to share if no-one else is using our tool

Preserve a HOL4 Identity

There is no point in chasing other systems.

Not all vectors of improvement point to positions occupied by existing systems.

If people want to use Isabelle/HOL or Coq, they should.

The Way Forward

- Identify (and then strengthen) the **U**nique **S**elling **P**oints
- Spend development time on important shortcomings
- Support existing users

What *Are* the USPs? (1)

The **HOL4 Tenets of Faith**:

1. Easy to write tools
2. Good documentation
3. Simple system
4. Stable APIs

Development mustn't endanger these.

What *Are* the USPs? (2)

Existing formalisations:

- hardware models
- CakeML
- probability
- ?

Clearly, we must commit to keeping these working

- and ensuring that owners want to keep developing them

Regression test process should help.

Important Shortcomings

Theory Mechanism:

- Theory files on disk should be pure data.
  - allowing manipulation by tools.

Fragile Proofs:

- implement declarative proof language?

Concurrency:

- use Isabelle's PIDE document-centric technology?

Tools:

- datatypes, HOLyHammer, ...?

Conclusion

HOL4 development will continue as long as people indicate they want to keep using the system...

Questions for the Audience

What do you think are the most **important** fixable shortcomings?

- *type abbreviation name spaces?*

What can be done to **improve** community?

- *how might we improve the website?*
- *what big attractive projects might we pursue?*

What **shouldn't** be changed?