Experiences from Exporting Proof Assistant Libraries

Michael Kohlhase, Florian Rabe

September 2022

Background

Our Proof Assistant Projects

OMDoc 2000–

- machine-oriented content representation language
- joint platform for MMT/LATIN content and OAF exports

MMT 2006–

- platform for building logical frameworks
- knowledge management infrastructure

LATIN 2008–2012, redesign 2020–

- uses MMT/LF to formalize "all" logical systems
- highly modular network of theories

OAF 2014–2020 and ongoing

- export proof assistant libraries relative to logic definitions in LATIN
  focus of this talk
- use for library integration

What we thought we were gonna do

Project Goals

Represent language of system S

- specification of syntax, semantics of S-logic
- represents all built-in/logical symbols $\rightarrow$, $\forall$, etc.
- manual in MMT using logical framework F

Export libraries of S

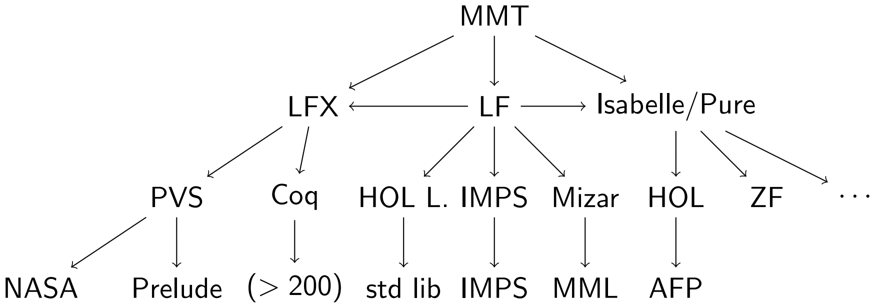
- port of libraries from S to MMT/F/S
- represents all user-defined/non-logical symbols
  definitions, theorems, etc.
- automated by instrumenting S

Coq, HOL Light, IMPS, Isabelle, Mizar, PVS

Integrate libraries

- now many systems and libraries in F
- systematically build system translations verifiable by F
- generic tool support via MMT/F proof checking, search, ...

Overview of OAF Libraries



Example: HOL Light Language

One declaration per primitive concept, logical symbol, rule

```
theory HOLLight : LF =
  holtype : type
  term    : holtype → type
  thm     : term bool → type

  bool    : holtype
  fun     : holtype → holtype → holtype
  Abs     : {A,B} (term A → term B) → term (A ⇒ B)
  Comb    : {A,B} term (A ⇒ B) → term A → term B
  equal   : {A} term A ⇒ (A ⇒ bool)

  REFL    : {A,X:term A} ⊢ X = X
  TRANS   : {A,X,Y,Z:term A}
    ⊢ X = Y → ⊢ Y = Z → ⊢ X = Z
 ...
```

MMT provides notation-based parsing, type reconstruction etc.

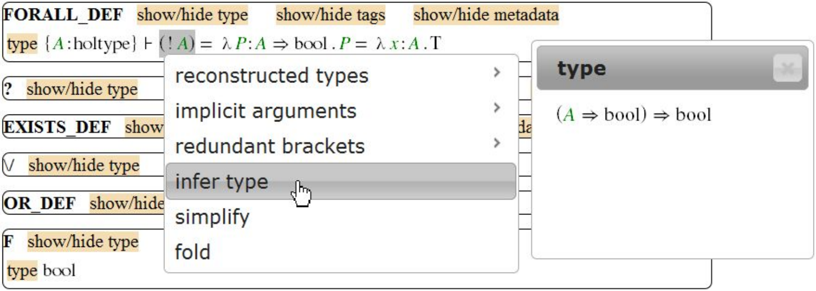
Example: HOL Light Library

Represent every statement in library as OMDoc XML, e.g.,

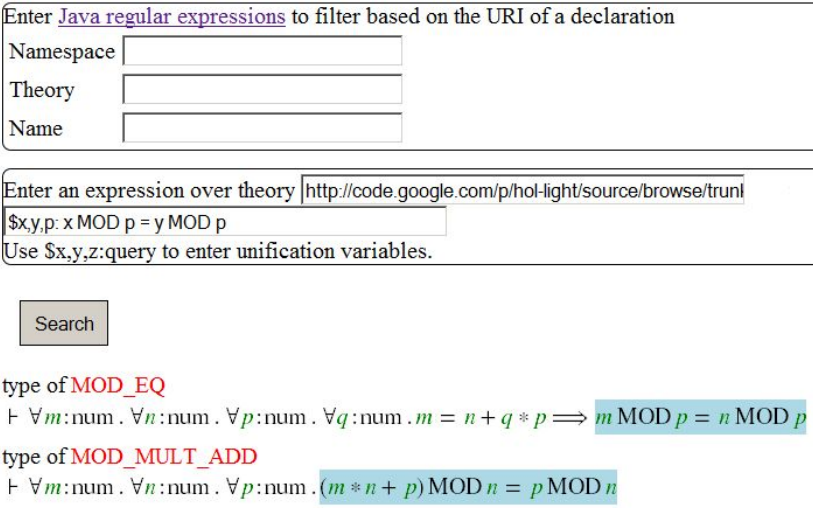
- HOLLight: `PRE: num -> num` in HOLLight
- MMT/LF: `PRE: term (fun num num)`
- OMDoc XML:

```
<constant name="PRE">
 <type>
   <OMA>
    <OMS module="LF" name="apply"/>
    <OMS module="HOLLight" name="term"/>
    <OMA>
      <OMS module="LF" name="apply"/>
      <OMS module="HOLLight" name="fun"/>
      <OMS module="nums" name="num"/>
      <OMS module="nums" name="num"/>
    </OMA>
  </OMA>
 </type>
</constant>
```

Example: HOL Light Services



Example: HOL Light Services



Enter Java regular expressions to filter based on the URI of a declaration

Namespace

Theory

Name

Enter an expression over theory http://code.google.com/p/hol-light/source/browse/trunk

$x,y,p: x MOD p = y MOD p

Use $x,y,z:query to enter unification variables.

Search

type of MOD_EQ

$\vdash \forall m:\mathrm{num}.\ \forall n:\mathrm{num}.\ \forall p:\mathrm{num}.\ \forall q:\mathrm{num}.\ m = n + q * p \Longrightarrow m \ \mathrm{MOD}\ p = n \ \mathrm{MOD}\ p$

type of MOD_MULT_ADD

$\vdash \forall m:\mathrm{num}.\ \forall n:\mathrm{num}.\ \forall p:\mathrm{num}.\ (m * n + p) \ \mathrm{MOD}\ n = p \ \mathrm{MOD}\ n$

Logic Translations

Most natural approach

- represent systems S, T in framework F
- formal translation $S \to T$ in F
- induces library translation

Evaluation

- perfect in theory
- works great for textbook logics — FOL, HOL, etc.
- rarely works for practical proof assistant logics — at best from simple to complex logics

Issues

- non-compositional translations — it's hard
- library integration problem — and it misses the point
  - incompatible types
  - incompatible definitions
  - incompatible subtyping

Issue: Non-compositional Feature Translation

Often features not directly expressible in target logic, e.g.,

- undecidable subtying of PVS
- soft types of Mizar
- partial functions of IMPS
- extensible records of PVS
- types non-empty of HOL, Mizar
- booleans+propositions of Coq
- universes of Coq, universes of Mizar
- theory inclusions of PVS

Non-compositional translations needed, which are often

- difficult/erroneous — often open research problems
- brittle — easily broken when system changes
- non-modular — may be broken in the presence of other features
- partial — fail on some rarely-used features

Issue: Library Integration Problem

Problem

- logic translation $S \to T$ induces library translation
- but yields copy of S-library that is unrelated to T-library

  T-library already contains definitions of some S-concepts

Alignments

- alignment = pair of S and T-symbol formalizing the same concept
- translation should respect alignments
- almost never the case out of the box   maybe not even for booleans

Note: Many "translations" in the literature are actually deep embeddings. These, by design, translate nothing to its counterpart, not even types or propositions.

Issue: Incompatible Types

aligned symbols may have different types, e.g.,

division by 0

- $div : num \to num \to num$ — total function with default value
- $div : num \to num \hookrightarrow num$ — partial function
- $div : num \to \{y : num | y \neq 0\} \to num$ — predicate subtype
- $div : num \to (y : num) \to \vdash y \neq 0 \to num$ — guard argument

semigroup on $(S, \circ, Assoc)$

- $theory\ SG\ \{S, \circ, Assoc, \ldots\}$ — plain theory
- $theory\ SG(S)\ \{\circ, Assoc, \ldots\}$ — carrier as parameter
- $theory\ SG(S, circ, Assoc)\ \{\ldots\}$ — all primitives as parameter
- $theory\ Magma\ \{S, circ, \ldots\}$, predicate $Assoc : Magma \to bool$
  axioms as separate predicate (needed with non-dependent records)

orthogonal choice: records vs. theories

orthogonal choice: extra parameter for universe of S

Issue: Incompatible Definitions

aligned symbols may have different definitions, e.g.,

often equivalent, but equivalence undecidable

The order $a \leq b$ in a lattice

- primitive, axiomatized
- $a \sqcap b = a$
- $a \sqcup b = b$

The type of real numbers

- axiomatic theory
- Dedekind cuts, Cauchy sequences, . . .
- intervals, computable reals

Issue: Incompatible Subtyping

alignments might not respect subtyping, e.g.,

number hierarchy $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$

- subsets, e.g., Mizar
- subtypes, e.g., in PVS
- separate types, e.g., HOL Light
- subsets on a type, e.g., HOL

algebraic hierarchy Magma, Semigroup, Monoid, . . .

- theories and inclusions, e.g., Isabelle, PVS
- extensible records with record subtyping, e.g., PVS
- separate records with forgetful functors, e.g., Coq

Library-Level Translations via Alignments

Idea

- logic translations alone not enough
- better: library translation via alignments needed

with enough alignments, logic translation somewhat optional

Problem: Alignments get complex

see issues above

- employ non-trivial inference, heuristics

  e.g., infer the non-zero proof for ternary division
- use alignments to get rough translation, then use target system to try to fill in gaps

  maybe machine learning/fuzzy parsers to fix minor syntax issues?

Problem: Managing large number of alignments

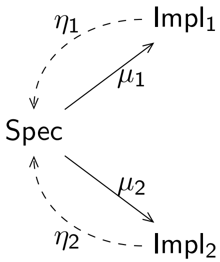
- manual or machine learning?
- star-shaped or system-pairwise?
- annotate inside libraries or in separate alignment database?

Library Integration via Interface Theories

Idea

- direct translations $S \to T$ often complex, difficult
  with or without alignments
- better (?): use axiomatic interface theories in the logical framework

- Spec: interface theory — e.g., Peano axioms, order, group
- Impl_i: implementation of Spec in system i
  e.g., built-in, inductive type, subtype of $\mathbb{R}$, ...
- μ_i: theory morphism witnessing how Spec is realized
- η_i: partial inverse of μ_i



Interface Theories: Pros and Cons

Appeal of interface theories

- abstract away prover/library idiosyncrasies
  forces integration-friendly definitions
- easy to write for non-experts
  most prover complexity only needed for doing proofs
- flexible choice of weakest possible logic typed FOL+X often enough

Challenges

- prover-independent library of interfaces needed
  big effort in addition to existing libraries
- coordinated community effort needed
- maybe contrary to recent trends in DTT-ITPs
  smart type system to interpret user input

What we ended up doing

Exports are hard

Only little work on library integration done

Exports are hard

- many time-consuming issues
  conceptual, logical, implementation, scalability, maintenance

Only little work on library integration done

Exports are hard

- many time-consuming issues
  conceptual, logical, implementation, scalability, maintenance
- intense collaboration with system experts necessary, e.g.,
  - Sacerdoti Coen's Coq XML export plugin separate paper
  - Wenzel subcontracted for 6-12 months to write Isabelle export
    MMT export now part of Isabelle release
  - Owre made numerous extensions to PVS on our request
  - Mizar completely redesigned XML export, partially with our feedback

Only little work on library integration done

Exports are hard

- many time-consuming issues
  conceptual, logical, implementation, scalability, maintenance
- intense collaboration with system experts necessary, e.g.,
  - Sacerdoti Coen's Coq XML export plugin separate paper
  - Wenzel subcontracted for 6-12 months to write Isabelle export
    MMT export now part of Isabelle release
  - Owre made numerous extensions to PVS on our request
  - Mizar completely redesigned XML export, partially with our feedback
- in fact, we stretched the project from 3 years to 6 years to get better exports

Only little work on library integration done

Exports are big

Language	Libraries	Modules	Decl.'s	RDF triples
PVS	Prelude+NASA	$1k$	$25k$	
Isabelle	distrib.+AFP	$12k$	$2M$	$40M$
HOL Light	Basic	200	$20k$	
Coq	> 50	$2k$	$200k$	$12M$
Mizar	MML	$1k$	$70k$	

Conceptual Issues

Elaboration

- high-level structure visible to users — much better for integration
- elaborated into low-level kernel structure
  often the only thing that can be exported
- Isabelle inductive types, Coq sections, Mizar definitions, ...
  disconnect can be massive but barely traced in tools

Handling of Abstract Theories

- aligning abstract theories critical for library integration
  ideal for working with interface theories
- vast diversity of language features and library conventions
  see above, very hard to bridge

Library compatibility

- delayed adaptation to new tool versions
- libraries of the same tool incompatible with each other
  e.g., Coq universe inference for *TYPE*

Logical Issues

Off-the-Shelf Logical Frameworks often not strong enough

- subtyping, quotient typing
- partial functions, undefinedness
- pattern-matching
- rewriting, computation as part of type system

Features that introduce names

- inductive types
- record types

Idiosyncratic module systems

- inferred type parameters in Isabelle locales
- imperative treatment of Coq sections
- undecidable set of identifiers in PVS includes

We designed and used extensions of LF with varying success.

Implementation issues

Tool internals complex

- often only understandable by system expert
- kernel hooks or kernel-generated files or export framework

Advanced logic features

- experimental — ongoing research, but used somewhere in library
- no or little documentation — reverse engineering not unusual

Inaccessible data

- elaboration data: high-level structure, relation to kernel structure
- comments
- source references: kernel data structures must link to source location

Scalability issues

Exports are big

- OMDoc on disk must be compressed
- internal structure sharing not alway exportable

  e.g., non-semantic sharing in Coq
- HTML presentation must be pre-generated

Nice LF representations are even bigger

- Coq representation in LF only via untyped term language

  $apply : term \to term \to term$
- typed representation explodes export size

  $apply : \Pi_{A,B}.term\, A \to term\, B \to term\,(A \Rightarrow B)$

Proof terms are even bigger

- some generated proofs cause export time outs
- especially bad in provers without built-in computation

  such as Isabelle rewriting (as opposed to Coq computation)
- dependency-only proofs in our exports

Maintenance issues

Human resources

- needed: coordinator+system expert+export implementer
- 2 weeks initial meeting+months of asynchronous work

Incentivization

- very challenging theoretically
- very labor-intensive
- but barely publishable

Decay

- export code decays
- system expert busy
- implementers move on
- language/tool changes
- tool improves in a way that deprecates export

Mizar: reimplemented from scratch after 10 years

What we think should be done going forward

Best Practice for an Export

Near the prover — implemented by system expert

- push button export of internal data structures all types inferred, etc.
- ad hoc XML schema or similar
- actively maintained by tool developers

documentation of XML schema as interface

Near MMT — implemented by us

- reads tool-near export
- uses MMT to generate OMDoc
- minor compositional translation to standardize abstract syntax
  no standardization of semantics

much improvement recently, e.g., Isabelle, Mizar, PVS

Proof Assistant Design

Embrace tool-near exports

- documented XML (or JSON or other) schema
- co-release exports with libraries
- re-read, re-check export — needed for testing

Design export-anticipating internal data structures

- source references
- pre- and post-elaboration structure, interconnected

Support integration-anticipating formalization

- abstract theories wherever possible
  records good for proving, bad for integration
- allow restricting logical strength
  formalize every theory in the weakest possible sublogic
- allow hiding the underlying logic as much as possible
  avoid heavy use of type system in basic definitions

High-Level Proof Interchange Language

Language

- Isar-like but not logic/tool-specific
- close to proof in math paper but with enough data to generate stubs for every prover

Prover integration

- every prover exports high-level proofs — relatively easy to do
- every prover tries to read high-level proofs

use alignment-based translation
heuristically fill gaps

Realization

- community commitment needed
- but very much in reach with little effort

Interface Theory Library

Proposal

- Community effort to build large library of interface theories
- Every prover community maintains alignments to their prover

Big challenge but

- QED Manifesto still applies — this would be a much easier goal
- similar to what FormalAbstracts already started building
- potential for major impact on wider scientific community

similar to what TPTP did for ATPs