Towards System Integration via a Math-in-the-Middle Ontology

Florian Rabe

Computer Science, University Erlangen-Nürnberg, Germany
LRI, University Paris-Sud, France

October 2018

Motivation

Problem: Islands of Incompatible Systems

Each formal math system needs huge investment to

- design the foundational system
- build scalable implementation
- build and verify a collection of formal definitions and theorems
- apply to practical problems

Each system fixes a logic and/or programming language

- type theories, set theories, logics, ...
  ACL2, Coq, HOL, Isabelle/HOL, Matita, Mizar, Nuprl, PVS,...
- computation systems, computer algebra systems ...
  Axiom, Sage, GAP, Maple, ...

systems mutually incompatible, communities very disjoint
contrast to traditional mathematics: foundation left implicit

Integration Long-standing Challenge Problem

QED manifesto written $>$ 20 years ago — still applies today

Goals

- Reuse across systems
  - exchange results: computations, proofs, database queries, documentation
  - integrate overlap across libraries
- Uniform entry point for outside users
  - comprehensive library browser
  - search across all libraries
  - standardized names, argument order, ...
  - link formal to informal libraries
- Generic tool support, e.g.,
  - user interface
  - search
  - hammering

Examples of Partial Successes

- ad hoc translations
  - between deduction systems, e.g., within HOL family
  - between computation systems, e.g., GAP-Sage

  not scalable to all systems
- logic standardizations
  - TPTP for (mostly) first-order logic ATPs
  - OpenTheory for HOL family

  no coverage of advanced logics, math domains
- import frameworks
  - Isabelle sledgehammer: delegation to specialized tools
  - Dedukti, ProofCert: proof checker
  - MathHub: uniform library repository
  - SageMath: integration of computation libraries
- export frameworks
  - FoCaLiZe: heterogeneous proof development
  - Why3: generate verification conditions, code

  star-shaped unidirectional topologies, not middleware

Math-in-the-Middle

Idea

Star-shaped architecture but

- no foundational commitment in the central library
- bidirectional (partial) library translations
- system-independent middleware for exchanging data

Middle library close to traditional mathematics

- existing standard that all formalizations already follow
- focus on formal interface theories for math domains
  - identifier name, arity, notation
  - types, axioms
  - examples, documentation

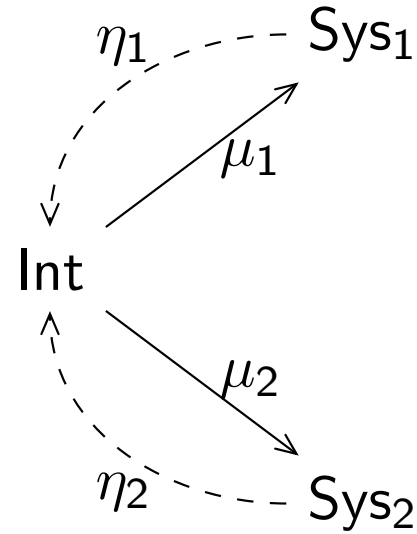
  the key information to mediate between systems/libraries
- not part of math-in-the-middle library
  - proof system
  - programming language
  - definitions

  would have to be foundation-specific

MitM-Based Library Integration

- Int: MitM interface theory
- Sys_i: library of system i
- μ_i: alignment describing how Sys_i realizes Int

  simplest case: set of pairs of identifiers
- η_i: partial inverse of μ_i does not cover proofs, programs



transfer of declarations and objects along edges

statically or dynamically

Example: Interface for Natural Numbers

Example formalizations in concrete systems

- primitive type
- inductive type
- defined in terms of other primitives, e.g., $0 := \varnothing$, $s(x) = x \cup \{x\}$
- subtype of integers (or other number sets)

MitM interface:

```
nat :   set
z :     nat
s :     nat → nat
plus :  nat×nat→set # _+_
```

```
s_def :       s(x)=x+s(z)
p_z_right :   x+z = x
p_z_left :    z+x = x
p_s_right :   x+s(y)=s(x+y)
p_s_left :    s(x)+y=s(x+y)
```

Not in interface: definitions, proofs, dependencies

Subtleties

Requirements for MitM language: everything Gilles said and

- soft typing
  - untyped foundation only canonical system-independent choice
  - typing best way to specify valid inputs
- subtyping: inherent feature of common data structures
  - subtyping or coercions?
  - decidable or undecidable?
  - how to support operators with multiple types?

e.g., plus on nat, int

Not the language of any existing system

build it in frameworks like MMT, Dedukti

My Work So Far

MMT Framework

Foundation-independent formal framework

12 years, $\approx 100,000$ loc, ≈ 500 pages of publications

`http://uniformal.github.io/`

- Flexible logical framework

  primitives capture common structure of formal systems

  LF, Isabelle, Dedukti recovered as MMT fragments
- Scalable library representation format
- Generic implementation of
  - Soundness-critical algorithms

    module system, type reconstruction, ...
  - Knowledge management services

    search, build system, library, ...
  - User-facing applications IDE, library browser, graph viewer, ...

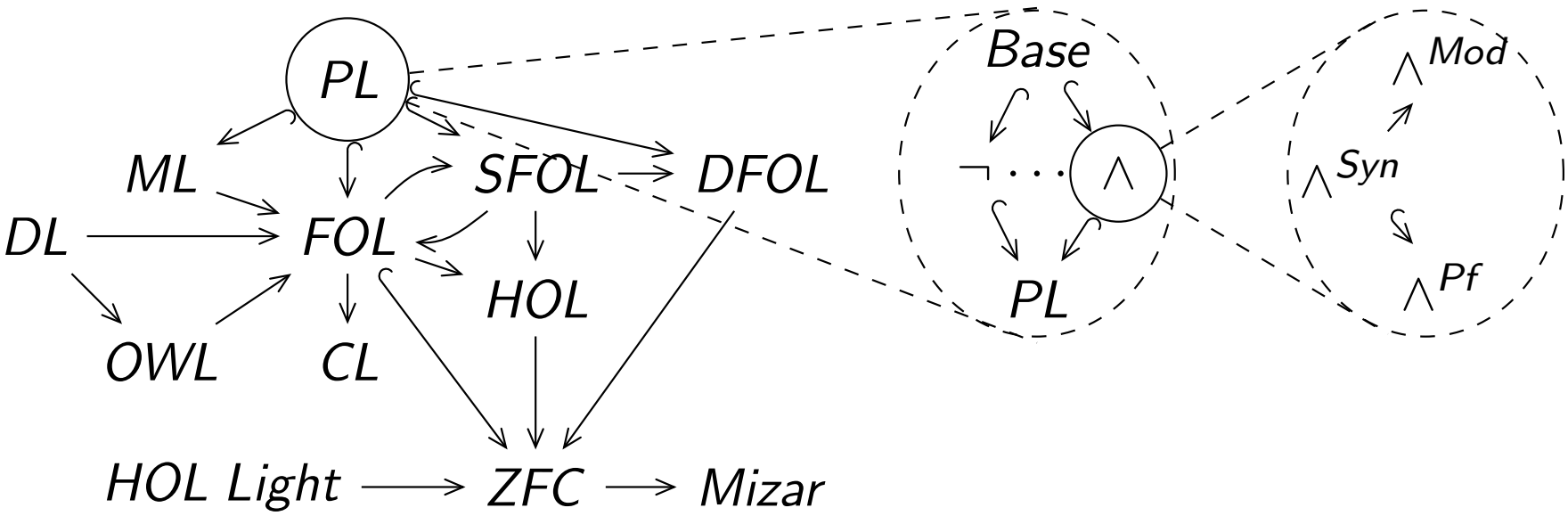
Ideal for defining MitM language and interfaces

`https://uniformal.github.io/`

LATIN Logic Atlas

- Logic interfaces written in MMT/LF
- Highly modular reference catalog of existing logics
- 2010–2012 and beyond, $\sim$ 1000 modules

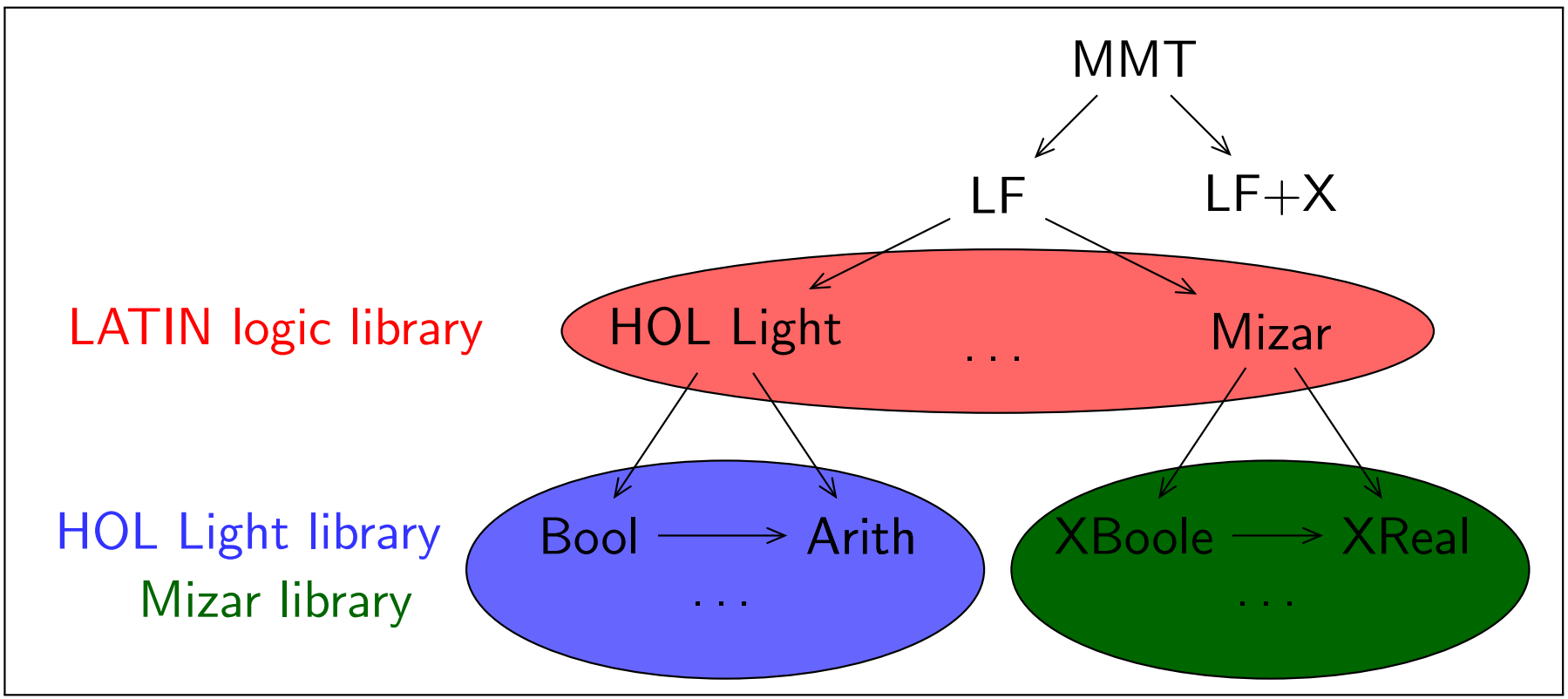
with Kohlhase, Mossakowski



https://gl.mathhub.info/MMT/LATIN

MathHub Archive of Formalizations

- OAF project 2014–2018: deduction systems
  with Kohlhase and external collaborators
- representation of formal libraries in uniform format
  Mizar, HOL Light, IMPS, Coq/Matita, PVS, Isabelle, . . .



https://gl.mathhub.info/

OpenDreamKit

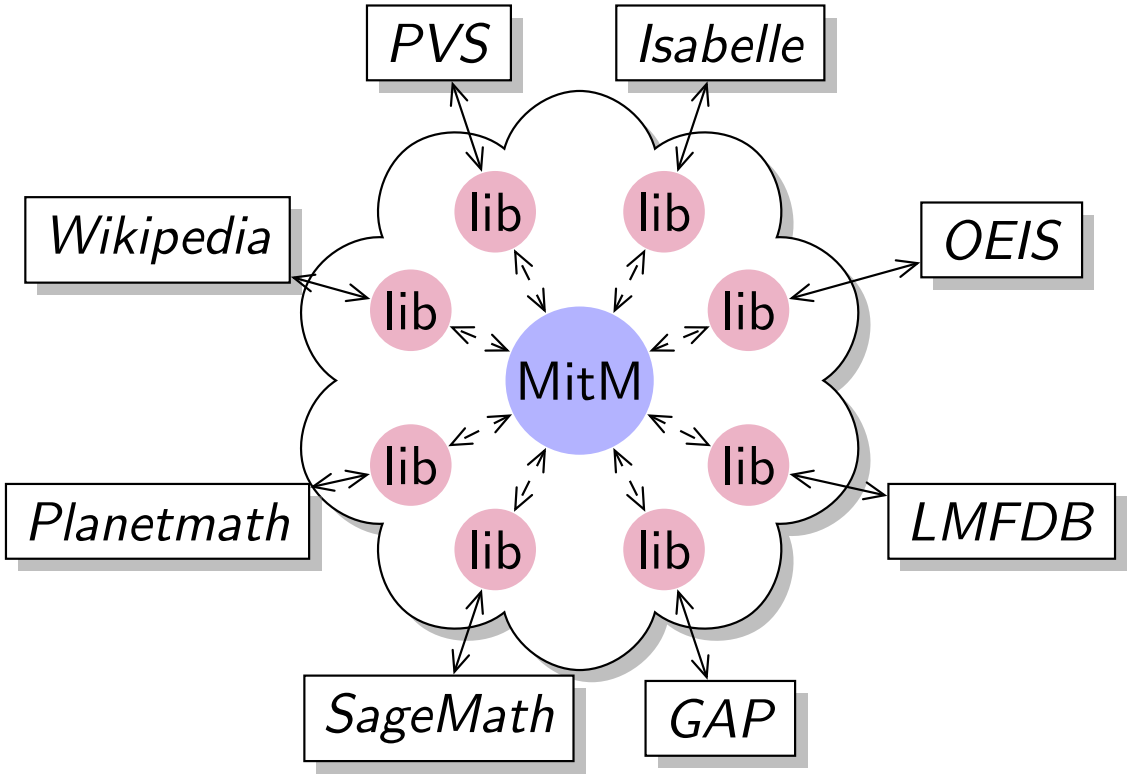
- EU math infrastructure project, 15 sites, 25 partners
- integrate mathematical data and computation
- MMT interfaces to
  - computation systems: SageMath, GAP, ...
  - databases: LMFDB, OEIS, ...
- MitM-interfaces for relevant mathematical data structures
- $\approx$ 200 MitM interface theories and hundreds of alignments from arithmetics, algebra, calculus, numerics, graph theory, topology, elliptic curves, ...
- MMT as MitM-middleware for exchanging objects

https://opendreamkit.org/

Practical Integration

Architecture

all system libraries represented (lib) in central system, e.g., MMT



Example: in SageMath, get Hecke number fields of Hilbert modular forms with degree 2

1. MMT queries LMFDB and represents result in MitM-language
2. MMT performs simple MitM-operations to build number fields
3. MMT returns result in SageMath abstract syntax

Joint Future Project

Let's

- build universal MitM library as large collaborative project
- use it to integrate across systems

Still challenging

- build library of MitM interfaces
- align MitM interfaces with individual system libraries

But much simpler than qed project or any single proof assistant

- most complexity in concrete systems mostly needed to prove/program
- we have the skills, tools, and manpower now to build MitM

a common project that still allows all groups to retain their system