The Future of Logic: Foundation-Independence

Florian Rabe

Jacobs University Bremen, Computer Science

World Congress on Universal Logic, June 27 2015

A Simplistic History of Logic

Antiquity	informal logic, Aristotle, Avicenna
	knowledge and reasoning are fundamental to science
1879	Frege, formal logic
1883	Cantor, naive set theory
1889	Peano axioms
	formality allows stronger applications
1901	Peano, Russell, paradoxa
1908, 1913	Russell, Whitehead, type theory
1908, 1922	Zermelo, Fraenkel, axiomatic set theory
	exact choice of formal language matters
1920s	Hilbert, reduction of truth to effective means
1929, 1936	Gödel, Gentzen, predicate logic
1931	Gödel, incompleteness
	there is no single best logic

Logic in Computer Science

- Tumultuous time also marks birth of computer science
  vision of mechanizing logic
- Competition between multiple logics
  - axiomatic set theory: ZF(C), GBvN, ...
  - λ-calculus:
    - typed or untyped
    - Church-style or Curry-style
  - new types of logic modal, intuitionistic, paraconsistent ,...
- Diversification into many different logics
  - fine-tuned for diverse problem domains
    far beyond predicate calculus
  - bridging gap between logic and programming languages
  - deep automation support
    decision problems, model finding, proof search, ...
- Economy of scale through computer processing

Selected Major Successes

Verified mathematical proofs

- 2006–2012: Gonthier et al., Feit-Thompson theorem
  170,000 lines of human-written formal logic
- 2003–2014: Hales et. al., Kepler conjecture (Flyspeck)
  $> 5,000$ processor hours needed to check proof

Software verification

- 2004–2010: Klein et al., L4 micro-kernel operating system
  390,000 lines of human-written formal logic
- since 2005: Leroy et al., C compiler (CompCert) 90% verified so far

Logic-based Artificial intelligence

- since 1984: Lenat et al., common knowledge (CyC)
  2 million facts in public version
- since 2000: Pease et. al., foundation ontology (SUMO)
  $25,000$ concepts

Future Challenges

Huge potential, still mostly unrealized

Applications must reach much larger scales

- software verification successes dwarfed by practical needs
  internet security, safety-critical systems, ...
- automation of math barely taken seriously by mathematicians

Applications must become much cheaper

- mostly research prototypes
- usually require PhD in logic
- tough learning curve
- time-intensive formalization

Two Formidable Bottlenecks

Each system requires $\approx$ 100 person-year investment to

- design the foundational logic
- implement it in a computer system
- build and verify a collection of formal definitions and theorems

  e.g., covering undergraduate mathematics
- apply to practical problems

human resource bottleneck

New scales brought new challenges

- no good search for previous results

  reproving can be faster than finding a theorem
- no change management support

  system updates often break previous work
- no good user interfaces — far behind software engineering IDEs

knowledge management bottleneck

The Dilemma of Fixed Foundations

Each system fixes a foundational logic

- Many systems
  ACL2, Coq, HOL, Isabelle/HOL, Matita, Mizar, Nuprl, PVS,. . .
  with different foundational logics
  type theories, set theories, first-order logics, higher-order logics, . . .
- Each system's results depend on fixed foundation
  contrast to mathematics: foundation left implicit
- All systems mutually incompatible

Exacerbates the other bottlenecks:

- Human resource bottleneck
  - no reuse across systems
  - very slow evolution of systems
- Knowledge management bottleneck
  - retrofitting to fixed foundation systems very difficult
    can be easier to restart from scratch
  - best case scenario: duplicate effort for each system

Example Problems

Collaborative QED Project, 1994

- high-profile attempt at building single library of formal mathematics
- failed partially due to disagreement on foundational logic

Voevodsky's Homotopy Type Theory, since 2012

- high-profile mathematician interested in applying logic
- his first result: design of a new foundation

Multiple 100 person-year libraries of mathematics

- developed over the last ~ 30 years
- overlapping but mutually incompatible   major duplication of efforts
- translations mostly infeasible

Hales's Kepler Proof

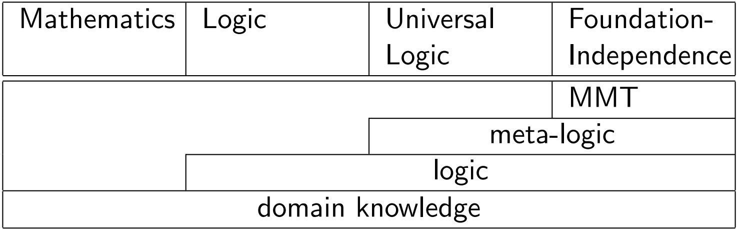
- distributed over two separate implementations of the same logic
- little hope of merging

My Vision: MMT as a Universal Logical Framework

MMT = meta-meta-theory/tool

a universal framework for the
formal representation of all knowledge and its semantics
in math, logic, and computer science

- Avoid fixing foundations wherever possible
- Obtain foundation-independent results . . .
- . . . and instantiate them for different foundations
- Use formal meta-logics in which to define logics . . .
- . . . and avoid fixing even the meta-logic



Overview

MMT language

- prototypical formal logic
- admits concise representations of most logics
- continuous development since 2006 (with Michael Kohlhase)
- > 200 pages of publication

MMT system

- API and services
- continuous development since 2007 (with > 10 students)
- $> 30,000$ lines of Scala code
- ~ 15 papers on individual aspects

Small Scale Example (1)

Meta-Logics in MMT

```
theory LF {
   type
   Pi        # Π V1 . 2                        name[:type][#notation]
   arrow     # 1 → 2
   lambda    # λ V1 . 2
   apply     # 1 2
}
```

Logics in MMT/LF

```
theory Logic: LF {
   prop : type
   ded  : prop → type # ⊢ 1                    judgments-as-types
}
theory FOL: LF {
  include Logic
  term         : type                          higher-order abstract syntax
  forall       : (term → prop) → prop #  ∀ V1 . 2
}
```

Small Scale Example (2)

FOL from previous slide:

```
theory FOL : LF {
  include Logic
  term          : type
  forall        : (term → prop) → prop  #  ∀ V1 . 2
}
```

Algebraic theories in MMT/LF/FOL:

```
theory Magma : FOL {
  comp : term → term → term # 1 ∘ 2
}
theory SemiGroup            : FOL {include Magma, ...}
theory CommutativeGroup : FOL {include SemiGroup, ...}
theory Ring : FOL {
  additive: CommutativeGroup
  multiplicative: Semigroup
  ...
}
```

Large Scale Example: The LATIN Atlas

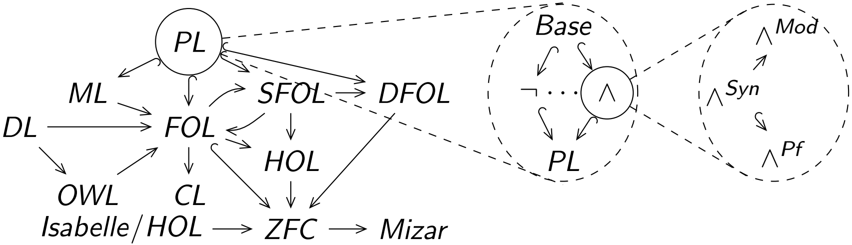
- Highly modular network of formal logics
  - propositional, common, modal, description, linear, unsorted/sorted first-order, higher-order, ...
  - ZF(C), category theory, ...
  - λ-calculi, product types, union types, ...

  and translations, e.g.,
  - typed to untyped
  - modal to first-order
  - classical to intuitionistic
  - type theory to set theory
  - propositions-as-types (Curry-Howard)
- Written in MMT/LF
- 4 years, with $\sim$ 10 students, $\sim$ 1000 modules

Large Scale Example: The LATIN Atlas (2)

An example fragment of the LATIN logic diagram

- nodes: MMT/LF theories
- edges: MMT/LF theory morphisms



- each node L is root for library MMT/LF/L
- each edge yields library translation functor

Design Cycle

- MMT arises by iterating the following steps
  1. Choose a typical problem
  2. Survey and analyze the existing solutions
  3. Differentiate between foundation-specific and foundation-independent concepts/problems/solutions
  4. Integrate the foundation-independent aspects into MMT
  5. Define interfaces to supply the foundation-specific aspects
- Separation of concerns between
  - foundation-independent framework
  - generic logical algorithms
  - generic knowledge management
  - customization with specific foundational logics

yields rapid prototyping for logic systems

Design Cycle

- MMT arises by iterating the following steps
  1. Choose a typical problem
  2. Survey and analyze the existing solutions
  3. Differentiate between foundation-specific and foundation-independent concepts/problems/solutions
  4. Integrate the foundation-independent aspects into MMT
  5. Define interfaces to supply the foundation-specific aspects
- Separation of concerns between
  - foundation-independent framework
  - generic logical algorithms
  - generic knowledge management
  - customization with specific foundational logics

  yields rapid prototyping for logic systems
- But how much can really be done foundation-independently?

  MMT shows: not everything, but a lot

Representation Language

- MMT theories uniformly represent
  - logics, set theories, type theories, algebraic theories, ontologies, ...
  - module system: state every result in smallest possible theory

    Bourbaki style applied to logic
- MMT theory morphisms uniformly represent
  - extension and inheritance
  - semantics and models
  - logic translations
- MMT objects uniformly represent
  - functions/predicates, axioms/theorems, inference rules, ...
  - expressions, types, formulas, proofs, ...
- Reuse principle: theorems preserved along morphisms

What are Logics, Translations, and Combinations?

- MMT allows coherent formal answers to previous contest questions
  "How to identify, translate, and combine logics?", Journal of Logic and Computation, 2014
- Logics are MMT theories
- Foundations are MMT theories e.g., ZFC set theory
- Semantics is an MMT theory morphism
  e.g., from FOL to ZFC
- Logic translations are MMT theory morphisms
- Logic combinations are MMT colimits

Logical Algorithms

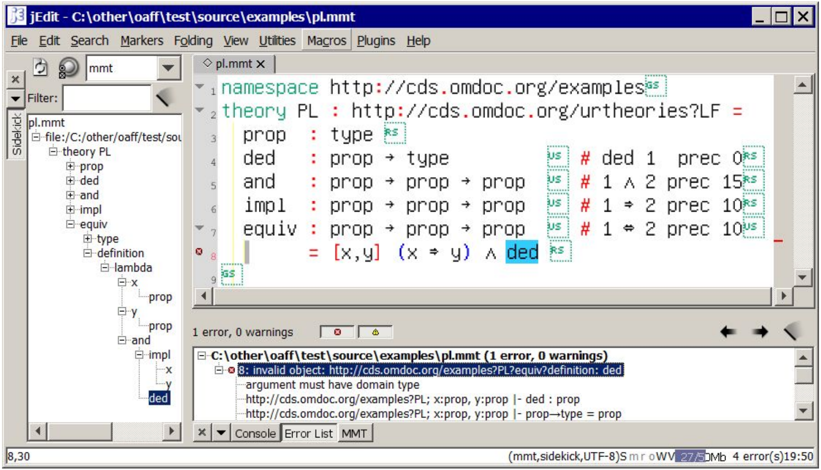
- Module system
  modularity transparent to foundation developer
- Concrete/abstract syntax
  notation-based parsing/presentation
- Type inference
  foundation plugin supplies core rules
- Interpreted symbols, literals
  integrates computation with logic
- Simplification
  combines computation and symbolic rewriting
- Theorem proving? probably (ongoing)

Knowledge Management

- Change management — recheck only if affected
- Project management — indexing, building
- Search — e.g., find all formulas of the form $A \vee \neg A$
- Querying — semantic web–style database
- Import from different foundations
- Export into non-logical formats
  programming languages, SVG graphs, LaTeX, HTML, ...

IDE for Efficient Formalization

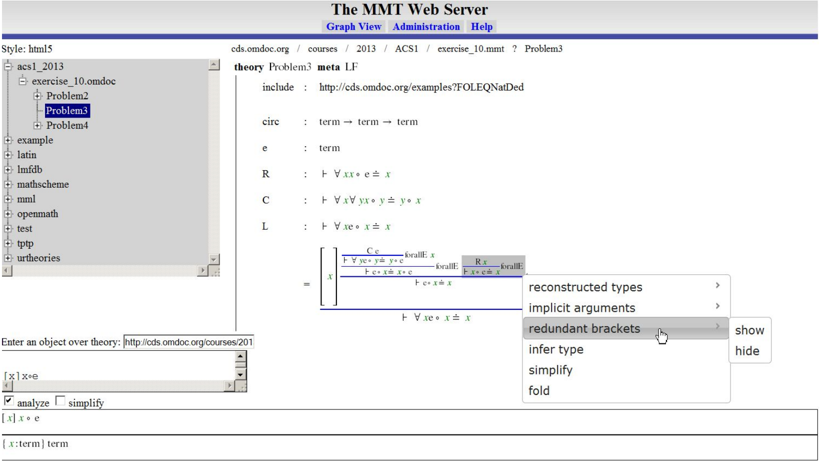
- Inspired by programming language IDEs
  hyper-links, interactivity, context-sensitive suggestions, . . .
- Modern text editor with MMT plugin



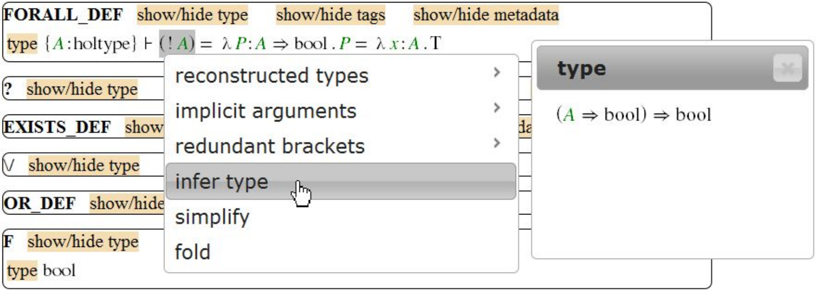
Interactive Library Browser

MMT content presented as HTML5+MathML pages

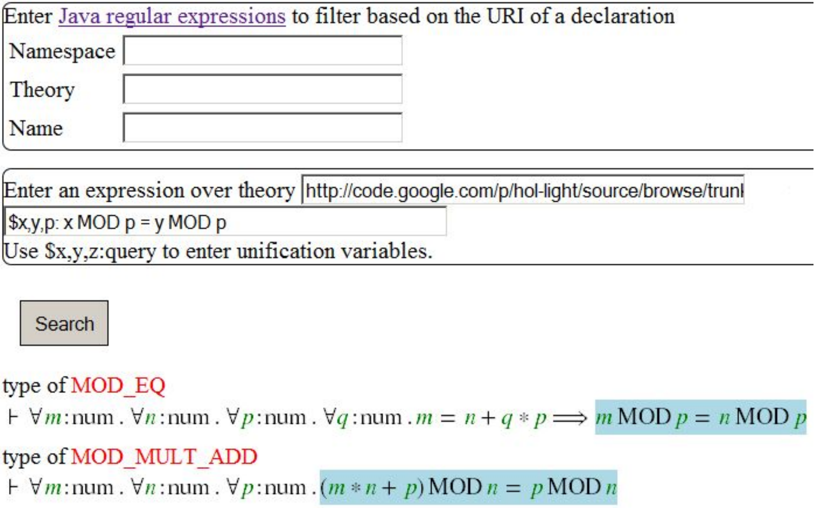
dynamic display, definition lookup, graph view, . . .



Browser Features: Type Inference



Browser Features: Search



LaTeX Integration

- MMT parses and checks LaTeX formulas
- MMT adds hyper-links, tooltips, inferred arguments into pdf
- upper part: LaTeX source for the item on associativity
- lower part: produced pdf with inferred type argument M

```
\begin{mmtscope}
 For all \mmtvar{x}{in M},\mmtvar{y}{in M},\mmtvar{z}{in M}
 it holds that !(x * y) * z = x * (y * z)!
\end{mmtscope}
```

A *monoid* is a tuple $(M, \circ, e)$ where

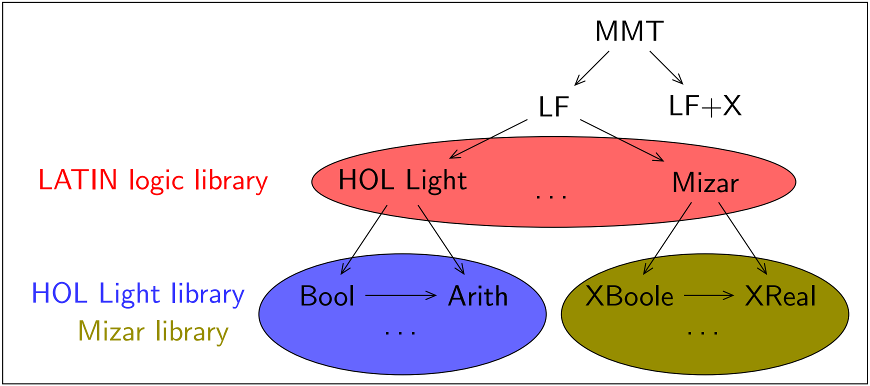
- M is a sort, called the universe.
- $\circ$ is a binary function on M.
- e is a distinguished element of M, the unit.

such that the following axioms hold:

- For all x,y,z it holds that $(x \circ y) \circ z =_M x \circ (y \circ z)$
- For all x it holds that $x \circ e =_M x$ and $e \circ x =_M x$.

Library Integration

- OAF: Open Archive of Formalizations open PhD position!
  Michael Kohlhase and myself, 2014-2017
- Goal: archival, comparison, integration of formal libraries
  Mizar, HOL systems, IMPS, Coq/Matita, PVS, . . .
- MMT as standardized interface language



Semi-Formal Multilingual Mathematical Glossary

- Collect real mathematical definitions

  Kohlhase and others, 2013, ongoing
- Mixes formal logic and informal mathematics
- Written by mathematicians from multiple fields
- Translated by students
- $\sim$ 1000 entries so far
- Uses MMT as background representation language

  integrates MMT with natural language
- Translations are semi-formal MMT theory morphisms

Semantic Alliance System

Goal: enrich domain-specific applications with logic-based services

- spreadsheets — Hutter and Kohlhase, 2012
- computer-aided design (CAD) — Kohlhase and Schröder, ongoing

Uses MMT as integration layer

- background knowledge formalized in MMT
- Semantic Alliance system integrates into Excel, AutoCAD etc.
- uses MMT to share knowledge across applications

Example:

- specification of screws in logic
- use logical reasoning to choose appropriate screws in CAD system
- use vendor/ordering information provided by spreadsheets

Virtual Research Environments for Mathematics

- OpenDreamKit project 2015-2019 open PhD positions!

  EU project, 11 sites, 25 partners

  http://opendreamkit.org/
- Support full life-cycle
  - exploration
  - proof and publication
  - archival and sharing of data and code
- Key requirements
  - allow using any foundation
  - allow abstraction from specific foundations

  just like mathematics does it
- MMT used as mediating system to integrate
  - formal mathematical logic
  - mathematical computation and data
  - informal mathematics and document preparation

Conclusion

- The future of logic: major scale-up at much lower costs

  foundation-independence is the key
- MMT arises by systematically building a foundation-independent framework
- Demonstrated success
  - foundation-independent representation language
  - mature implementation
  - easy to instantiate with specific foundations

    rapid prototyping logic systems
  - collection of deep foundation-independent results
  - collection of major MMT-based applications
- Particularly interesting for
  - areas with little automation support
  - areas with new, changing foundations
  - integration/combination of logics and systems