

MMT is **foundation-independent**:

1. Developer defines new logic
2. MMT yields complete MKM system for it

MMT is **application-independent**:

- ▶ No single MMT application
- ▶ Instead: focus on data model, interfaces, generic services
- ▶ Logical: parsing, type-checking, module system, ...
- ▶ MKM: change management, querying, presentation, ...
- ▶ Applications developed on top

Formal editing: [jEdit-MMT](#)

navigation, hyperlinking,  
auto-completion, tooltips, ...

Narrative editing: [LaTeX-MMT](#)

formulas processed by MMT –  
type-checking, cross-references, ...

Browsing: [MMT web server](#)

definition lookup, type inference  
theory graphs, ...

Building: [MMT scripting language](#)

easy import/export interfaces  
MMT services as build tasks

# The MMT API: A Generic MKM System

Florian Rabe

Jacobs University Bremen

CICM System Description (S&P Track), July 2013

See <https://trac.kwarc.info/MMT> for papers, source code, etc.

# MMT Overview

- ▶ Universal framework for formal mathematical/logical content
- ▶ Continuous development since 2007 [5]
- ▶ Close relatives
  - ▶ logical frameworks like LF, Isabelle  
but: even more generic, knowledge management, more system integration
  - ▶ OMDoc/OpenMath  
but with formal semantics, more automation
- ▶ MMT System:
  - ▶ MMT data structures
  - ▶ logical and knowledge management services
  - ▶ ~ 10 CICM papers on individual services
  - ▶ various individual applications utilizing the services
  - ▶ ~ 30,000 lines of Scala code total

# Central Idea: Foundation-Independence

1. We can fix and implement a logical theory      e.g., set theory
2. We can fix and implement a logic  
then **define many theories in it**      e.g., first-order logic
3. We can fix and implement a logical framework  
then **define many logics in it**      the foundation, e.g., LF
4. We can fix and implement a meta-framework  
then **define many logical frameworks in it**  
foundation-independence: MMT

# Highly Reusable Results through Foundation-Independence

- ▶ Conceptual
  - ▶ intuitions, documentation, teaching
  - ▶ definitions, meta-theorems
  - ▶ algorithms
- ▶ Knowledge management
  - ▶ editing, parsing
  - ▶ change management [2]
  - ▶ project management, distribution [1]
  - ▶ search, querying [4]
  - ▶ interactive browsing
  - ▶ system integration, semantic interfaces
- ▶ Logical
  - ▶ module system, namespace management [5]
  - ▶ type reconstruction
  - ▶ computation [3]

# MMT Design Methodology

1. Choose a typical problem
  - logical: e.g., type reconstruction, module system
  - MKM: e.g., change management, querying
2. Survey and analyze the existing solutions
3. Differentiate between **foundation-specific** and **foundation-independent** definitions/theorems/algorithms
4. Integrate the foundation-independent aspects into MMT
  - language and system
5. Define interfaces to supply the logic-specific aspects
  - formal theory and plugin interfaces
6. Repeat

# MMT Language Features

*few primitives ... that unify different domain concepts*

- ▶ Judgments as types, proofs as terms
  - unifies expressions and derivations
- ▶ Higher-order abstract syntax
  - unifies operators and binders
- ▶ Category of theories
  - unifies logical theories, logics, foundations
  - ▶ languages as theories
  - ▶ relations as theory morphisms
- ▶ Module system (little theories)
  - unifies inheritance and representation theorems
- ▶ Models as morphisms (categorical logic)
  - unifies syntactical translations and semantic interpretations

## A Small Formalization Example in MMT

The logical framework LF in MMT:

```
theory Types { type }
theory LF {include Types,  $\Pi$ ,  $\rightarrow$ ,  $\lambda$ ,  $@$  }
```

First-order Logic defined in MMT/LF:

```
theory Logic meta LF {o: type, ded : o  $\rightarrow$  type }
theory FOL meta LF {
  include Logic
  u: type. imp: o  $\rightarrow$  o  $\rightarrow$  o, ...
}
```

Algebraic theories in MMT/LF/FOL:

```
theory Magma meta FOL { o: u  $\rightarrow$  u  $\rightarrow$  u }
...
theory Ring meta FOL {
  additive: CommutativeGroup
  multiplicative: Semigroup
  ...
}
```



## Key Idea: Application-independence

1. No a-priori commitment to a particular application
2. Focus on API for MMT data model
3. Abstraction from physical storage  
MMT works with file systems, databases, remote servers
4. Abstraction from user interfaces  
MMT provides API, HTTP, shell interfaces
5. Advanced functionality as independent, reusable services  
parsing, type-checking, presentation, ...  
that are customized by plugins  
typing rules, presentation styles, ...

*Individual applications are built flexibly on top of the API.*

# Application Example: Editing

- ▶ IDE-like editor for MMT projects
- ▶ MMT acting as a plugin for jEdit text editor
- ▶ MMT handles data model aspects
  - ▶ parsing (based on plugins, notations)
  - ▶ type checking (based on plugins for typing rules)
  - ▶ auto-completion suggestions
- ▶ jEdit handles GUI aspects
  - ▶ outline view
  - ▶ error highlighting
  - ▶ hyperlinks (= click on operator, jump to declaration/definition)
  - ▶ context-sensitive auto-completion: show identifiers that
    - ▶ are in scope
    - ▶ have the right type

# Application Example: Editing

## Example feature: pop up shows reconstructed arguments

The screenshot shows a theorem prover interface with a proof script. The script defines a namespace, imports theories, and defines several logical rules and a test. A yellow highlight is placed on a line in the script, and a tooltip popup shows the reconstructed arguments for that line.

```
namespace http://cvs.ondoc.org/test-new.mmt
theory FOL : http://cvs.ondoc.org/urtheories7LF =
  bool : type
  univ : type
  and : bool → bool → bool
  impI : bool → bool → bool
  equiv : bool → bool → bool
  ded : bool → type

theory FOLProofs : http://cvs.ondoc.org/urtheories7LF =
  include "FOL.mmt"
  andI1 : (A,B) ded A → ded B → ded A ∧ B
  andE1 : (A,B) ded A ∧ B → ded A
  andE2 : (A,B) ded A ∧ B → ded B
  impI : (A,B) (ded A → ded B) → ded A → B
  impE : (A,B) ded A → B → ded A → ded B
  impI2 : (A,B)(C) (ded A → ded B → ded C) → ded A → (B → C)
  = (A,B,C) (f : impI (p) (impI (q) f p q))
  test : (A) ded A → (A ∧ A)
  = (impI B C) (impI B C) ded A → (A ∧ A)
```

The tooltip popup for the highlighted line shows the reconstructed arguments:

```
(impI B C) (impI B C) ded A → (A ∧ A)
```

# Application Example: Editing

Example feature: auto-completion shows only identifiers that are in scope and have the right type

The screenshot shows the Emacs editor with a Lean file named `test-new.mmt`. The left sidebar displays a project tree with a tree view of the file's structure, including sections for `theory FOL`, `theory FOL.Proofs`, and `test`. The main editor window shows the following code:

```
1 namespace http://cde.s.ondoc.org/test-new
2
3 theory FOL : http://cde.s.ondoc.org/untheories?LF =
4   bool : type
5   univ : type
6   and : bool → bool → bool
7   imp1 : bool → bool → bool
8   equiv : bool → bool → bool
9   ded : bool → type
10
11 theory FOL.Proofs : http://cde.s.ondoc.org/untheories?LF =
12   include ?FOL
13   andI : (A,B) ded A → ded B → ded (A ∧ B)
14   andE1 : (A,B) ded (A ∧ B) → ded A
15   andE2 : (A,B) ded (A ∧ B) → ded B
16   impI : (A,B) (ded A → ded B) → ded A → B
17   impE : (A,B) ded A → B → ded (A → B)
18   imp2I : (A,B)(C) (ded A → ded B → ded C) → ded A → (B → C)
19   |
20   | = (A,B,C) (f) impI (p) (impI (q) f p q)
21
22 test : (A) ded A → (A ∧ A)
23
```

At line 22, the text `(A) ded A → (A ∧ A)` is being edited. A dropdown menu of auto-completion suggestions is visible, listing identifiers from the current scope and their types:

- `FOL.Proofs.andE1` (type `A`)
- `FOL.Proofs.andE2` (type `A`)
- `FOL.Proofs.impI` (type `A`)
- `FOL.Proofs.impE` (type `A`)
- `FOL.Proofs.test` (type `A`)

The status bar at the bottom indicates the cursor is at line 24, column 45. The bottom right corner shows the file path `(mmt_sdelock_UTF-8)S mtr @WC Ab 2 error(s)17:32`.

# Application Example: Mathematical Documents

- ▶ Unified document format LaTeX-MMT
- ▶ Processed by LaTeX
- ▶ MMT-relevant aspects represented in special macros
  - ▶ `mmt.sty` sends them to MMT via HTTP during compilation
  - ▶ MMT returns LaTeX snippet, which LaTeX processes further
- ▶ MMT processing includes
  1. parsing
  2. type reconstruction
  3. generation of high-quality LaTeX that includes
    - ▶ reconstructed information
    - ▶ cross-references
    - ▶ tooltips

## Application Example: Mathematical Documents

- ▶ upper part:  $\text{\LaTeX}$ -MMT source for highlighted line
- ▶ lower part: pdf after compiling with  $\text{\LaTeX}$ -MMT
- ▶ enhanced LaTeX features generated by MMT
  - ▶ type argument  $M$  of equality symbol is inferred and added
  - ▶ cross-references from each symbol to its definition (works across pdfs)

---

```
\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  $e \circ x =_M x$ .

# Application Example: Interactive Browsing

- ▶ MMT API exposed through HTTP server
- ▶ Javascript/Ajax for interactive browsing of MMT projects  
e.g., definition lookup, dynamic type inference
- ▶ Interactive graph view based on SVG

document derived.omdoc

**remote module** FalsityExt

**remote module** NEGExt

**theory** IMPEExt **meta** lf

**include** IMP

  imp2I : ((ded A → ded B → ded C) → ded A imp (B imp C))

          = [f:ded A → ded B → ded C]impI ([p:ded A]impI ([q:ded B]f p q))

  imp2E : (ded A imp (B imp C) → ded A → ded B → ded C)

          = [p:ded A imp (B imp C)][q:ded A][r:ded B]impE (impE p q) r

**remote module** CONJExt

**remote module** DISJExt

**remote module** Equiv

type

ded A imp (B imp C)

infer type  
reconstructed types  
implicit arguments  
implicit binders  
redundant brackets  
Fold

# Application Example: Build Tool

MMT chains build steps on MMT projects

- ▶ Import **MMT native syntax, Twelf, Mizar, OWL, TPTP, ...**
- ▶ Computation
  - ▶ validation, type reconstruction
  - ▶ flattening
- ▶ Indexing
  - ▶ theory graph
  - ▶ document hierarchy
  - ▶ substitution tree of formulas (via MathWebSearch)
  - ▶ relational index (MMT ontology)
- ▶ Export  
**HTML+presentation MathML, SVG diagrams, Scala code [3], ...**



# Videos

- ▶ Editing (jEdit-MMT)
  - outline view, error highlighting, smart auto-completion
  - display of inferred information, hyperlinking

[http://youtu.be/tdd45Uv7j\\_g](http://youtu.be/tdd45Uv7j_g) (left QR code)
- ▶ Interactive browsing (MMT web server)
  - semantic highlighting, showing reconstructed information
  - dynamic type inference, navigation, theory graph display

<http://youtu.be/MPTzW86voI4> (right QR code)



- [1] F. Horozal, A. Iacob, C. Jucovski, M. Kohlhase, and F. Rabe.  
Combining Source, Content, Presentation, Narration, and Relational Representation.  
In J. Davenport, W. Farmer, F. Rabe, and J. Urban, editors, *Intelligent Computer Mathematics*, pages 212–227. Springer, 2011.
- [2] M. Iancu and F. Rabe.  
Management of Change in Declarative Languages.  
In J. Campbell, J. Carette, G. Dos Reis, J. Jeuring, P. Sojka, V. Sorge, and M. Wenzel, editors, *Intelligent Computer Mathematics*, pages 325–340. Springer, 2012.
- [3] M. Kohlhase, F. Mance, and F. Rabe.  
A Universal Machine for Biform Theory Graphs.  
In J. Carette, D. Aspinall, C. Lange, P. Sojka, and W. Windsteiger, editors, *Intelligent Computer Mathematics*, pages 82–97. Springer, 2013.
- [4] F. Rabe.  
A Query Language for Formal Mathematical Libraries.  
In J. Campbell, J. Carette, G. Dos Reis, J. Jeuring, P. Sojka, V. Sorge, and M. Wenzel, editors, *Intelligent Computer Mathematics*, pages 142–157. Springer, 2012.
- [5] F. Rabe and M. Kohlhase.  
A Scalable Module System.  
*Information and Computation*, 2013.  
to appear; see [http://kwarc.info/frabe/Research/RK\\_mmt\\_10.pdf](http://kwarc.info/frabe/Research/RK_mmt_10.pdf).