

# MMT Tutorial, Part 1: Designing Languages in MMT

Florian Rabe, Mihnea Iancu, Dennis Müller

Jacobs University Bremen

CICM 2016

Bringing your notebook is recommended but not required.

# Overview

1. Brief introduction to MMT
2. Download, install MMT and MMT IDE
3. Design languages in MMT
  - 3.1 LF as an example of a logical framework LF
  - 3.2 FOL as an example of a formal language
  - 3.3 algebraic theories as examples of domain knowledge
  - 3.4 module system for algebra
  - 3.5 design logics modularly
  - 3.6 implement logical frameworks modularly

## Further Resources

- ▶ MMT homepage: <http://uniformal.github.io/>
- ▶ Introductory articles: <http://uniformal.github.io/doc/philosophy/intros.html>
- ▶ Publications: <http://uniformal.github.io/doc/philosophy/papers.html>
- ▶ Sources: <http://uniformal.github.io/MMT>
- ▶ API documentation: <http://uniformal.github.io/apidoc>

# Language-Independence

MMT = meta-meta-theory/tool

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

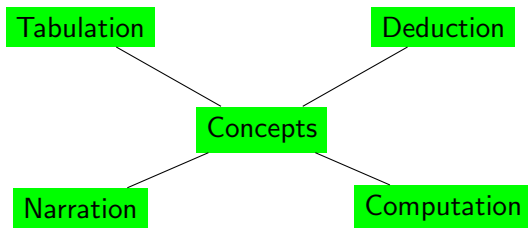
- ▶ Avoid fixing languages wherever possible
- ▶ Use formal meta-languages for defining languages ...
- ▶ ...and avoid fixing even the meta-languages.

Obtain (meta-)language-independent results

Mathematics	Formalization	Logical Framework	Language-Independence
			meta-meta-framework
			meta-language
			language (logics, DSLs, etc.)
			domain knowledge

# Subsume All Paradigms of Knowledge Representation

- ▶ Conceptualization: identifiers and their properties
- ▶ Narration: human-oriented, informal-but-rigorous
- ▶ Deduction: machine-verifiable, formal
- ▶ Computation: executable, algorithmic
- ▶ Tabulation: databases, queryable



# Design Principles

## Separation of concerns between

- ▶ language development logical primitives, rules
- ▶ knowledge management e.g., search, change management
- ▶ verification e.g., type checking, theorem prover
- ▶ application development e.g., IDE, proof assistant

## Universal language

- ▶ few primitives ...
- ▶ that unify different domain concepts

## Language-Independent Implementations

- ▶ possible for surprisingly many results
- ▶ yields rapid prototyping for logic systems

# Language-Independent Results So Far

## Knowledge Management

- ▶ Change management recheck only if affected
- ▶ Project management indexing, hosting
- ▶ Extensible build system presentation, import/export, ...
- ▶ Search, querying substitution-tree and relational index
- ▶ Browser interactive web browser
- ▶ Editing IDE-like graphical interface

## Logical Results

- ▶ Module system modularity transparent to foundation developer
- ▶ Concrete/abstract syntax notation-based parsing/presentation
- ▶ Type reconstruction foundation plugin supplies only core rules
- ▶ Simplification rule-based, integrated with type reconstruction
- ▶ Anticipated: Theorem proving, code generation, stateful computation

# Meta-Hierarchy

