

Rapid Prototyping Formal Systems in MMT: 5 Case Studies

Dennis Müller and Florian Rabe

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

June 2019

Motivation

Logical Frameworks

= meta-logic in which syntax and semantics of object logics are defined

Automath, LF, Isabelle

Advantages

- ▶ Universal concepts expressions, substitution, typing, equality, ...
- ▶ Meta-reasoning consistency, logic translations, ...
- ▶ Rapid prototyping type reconstruction, theorem proving, ...
- ▶ Generic tools theorem prover, module system, IDE, ...

Simplicity vs. expressivity

- ▶ Meta-logic must be simple to be scalable, trustworthy
- ▶ Object logic must be expressive to be practical
- ▶ Big challenge for frameworks

Designing Logical Frameworks

Typical approach:

- ▶ choose a λ -calculus
- ▶ add other features
 - ▶ logic programming (λ -Prolog)
 - ▶ meta logic (Twelf, Abella)
 - ▶ proof assistant for object logic (Isabelle)
 - ▶ concurrency (CLF)
 - ▶ reasoning about contexts (Beluga)
 - ▶ rewriting (Dedukti)
 - ▶ external side conditions (LLFP)
 - ▶ coupling with proof-assistant support (Hybrid)
 - ▶ user-defined unification hints (ELPI)
 - ▶ ...

Problems

- ▶ Divergence due to choice of other features
- ▶ Even hypothetical union not expressive enough for real-life logics
no way to define, e.g., HOL Light, Mizar, PVS

Experimentation with Formal Systems

Customize the system fundamentals

- ▶ increasingly complex problem domains
e.g., mathematics, programming languages
- ▶ plain formalization introduces too many artifacts to be human-readable
- ▶ therefore: allow users to define how to interpret human input
e.g., custom parsing, type reconstruction

Examples:

- ▶ unification hints (Coq, Matita)
 - ▶ extra-logical declarations
 - ▶ allow users to guide incomplete algorithms (e.g., unification)
- ▶ meta-programming (Idris, Lean)
 - ▶ expose internal datatypes to user
 - ▶ allow users to program extensions in the language itself

MMT = Meta-Meta-Theory/Tool

Problem:

- ▶ logical frameworks not expressive for practical logics
- ▶ more system experimentation needed
- ▶ trend towards fine-grained user control

Foundation-independence: use logical frameworks without committing to a specific one

Mathematics	Logic	Logical Frameworks	Foundation-Independence
			MMT
		logical frameworks	
	logic, programming language, ...		
domain knowledge			

The UniFormal Library

Large Scale Example: The LATIN Atlas

- ▶ DFG project 2009-2012 (with DFKI Bremen and Jacobs Univ.)
- ▶ Highly modular network of little logic formalizations
 - ▶ separate theory for each
 - ▶ connective/quantifier
 - ▶ type operator
 - ▶ controversial axioms e.g., excluded middle, choice, ...
 - ▶ base type
 - ▶ reference catalog of standardized logics
 - ▶ documentation platform
- ▶ Written in MMT/LF
- ▶ 4 years, with ~ 10 students, ~ 1000 modules

The LATIN Atlas of Logical Systems

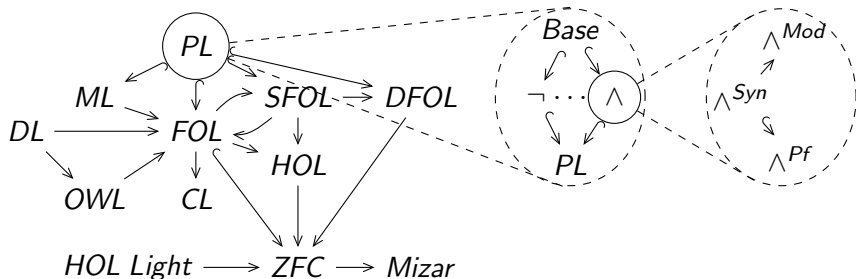
The LATIN Atlas is huge: That's me pointing at the theory for first-order logic



Logic Diagrams in LATIN

An example fragment of the LATIN logic diagram

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

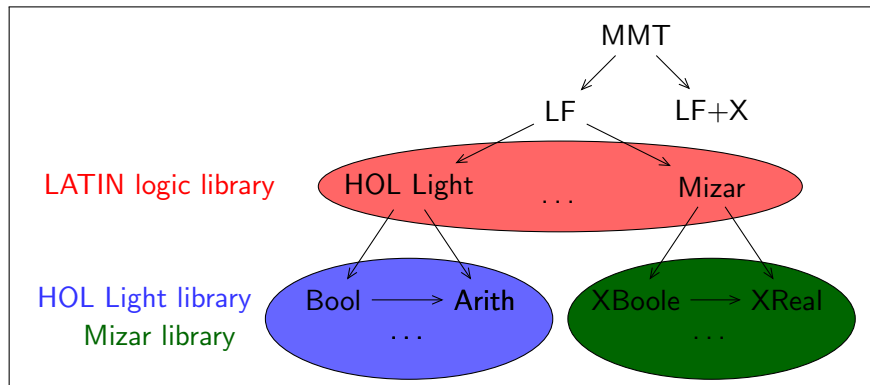


- ▶ each node is root for library of that logic
- ▶ each edge yields library translation functor

library *integration* very difficult though

OAF: Integration of Proof Assistant Libraries

- ▶ DFG project, 2014–2020, 15 contributors
- ▶ Big, overlapping libraries joined in MMT as the uniform representation language
 - > 100 GB XML in total
 - Mizar, HOL systems, IMPS, Coq, PVS, Isabelle...
- ▶ enables archival, comparison, integration



OpenDreamKit: Virtual Math Research Environments

- ▶ EU project, 2015-2019, 15 sites, 25 partners
<http://opendreamkit.org/>
- ▶ MMT as mediator system
 - ▶ system-independent formalization of math > 200 theories
no proofs, no algorithms
 - ▶ integration of math computation systems
SageMath, GAP, Singular: services interfaces defined in MMT
 - ▶ ... and math databases
LMFDB, OEIS: database schemas defined in MMT

Example: dynamic retrieval

- ▶ SageMath user needs 13th transitive group with conductor 5
- ▶ SageMath queries MMT
- ▶ MMT retrieves it from LMFDB, translates it to SageMath syntax

MathHub

GitHub-like but for MMT projects <https://gl.mathhub.info>

- ▶ 251 Repositories
- ▶ 187 Users
- ▶ 28.5 GB in March, probably doubled by now

For example:

Language	Library	Modules	Declarations
MMT	Math-in-the-Middle	220	826
LF	LATIN	529	2,824
PVS	Prelude+NASA	974	24,084
Isabelle	Distribution+AFP	9553	1,472,280
HOL Light	Basic	189	22,830
Coq	> 50 in total	1,979	167,797
Mizar	MML	1,194	69,710
SageMath	Distribution	1,399	
GAP	Library		9,050

MMT Overview

Basic Concepts

Design principle

- ▶ few orthogonal concepts
- ▶ uniform representations of diverse languages

sweet spot in the expressivity-simplicity trade off

Concepts

- ▶ theory = named set of declarations
 - ▶ foundations, logics, type theories, classes, specifications, ...
- ▶ theory morphism = compositional translation
 - ▶ inclusions, translations, models, katamorphisms, ...
- ▶ constant = named atomic declaration
 - ▶ function symbols, theorems, rules, ...
 - ▶ may have type, definition, notation
- ▶ term = unnamed complex entity, formed from constants
 - ▶ expressions, types, formulas, proofs, ...
- ▶ typing $\vdash_{\mathcal{T}} s : t$ between terms relative to a theory
 - ▶ well-formedness, truth, consequence ...

Example: Propositional Logic in the MMT IDE

The screenshot shows the MMT IDE interface. On the left is a file browser showing the project structure, including a 'theory PL' folder. The main editor window displays the following MMT code:

```

namespace http://cde.ondoc.org/examples

// @_title Propositional Logic in MMT
// @_author Florian Rabe

/T
Intuitionistic propositional logic with natural deduction rules and a few example proofs

theory PL : ur:?LF =

# :types The Basic Concepts

/T the type of propositions
prop : type

# Constructors

/T The constructors provide the expressions of the types above.

and  : prop → prop → prop | # 1 ∧ 2 prec 15
impl : prop → prop → prop | # 1 * 2 prec 10

/T Equivalence is defined such that for [F:prop,G:prop] we define $F*$G$ as $(F * G) ∧ (G * F)$.
equiv : prop → prop → prop | # 1 * 2 prec 10
      = [x,y] (x = y) ∧ (y = x)
  
```


Small Scale Example (1)

Logical frameworks in MMT

```

theory LF {
  type
  Pi      #  $\Pi V1 . 2$                                 name[: type][#notation]
  arrow   #  $1 \rightarrow 2$ 
  lambda  #  $\lambda V1 . 2$ 
  apply   #  $1\ 2$ 
}

```

Logics in MMT/LF

```

theory Logic : LF {
  prop : type
  ded   : prop  $\rightarrow$  type #  $\vdash 1$                                 judgments-as-types
}
theory FOL : LF {
  include Logic
  term      : type                                higher-order abstract syntax
  forall    : (term  $\rightarrow$  prop)  $\rightarrow$  prop #  $\forall V1 . 2$ 
}

```

Small Scale Example (2)

FOL from previous slide:

```

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

```

Proof-theoretical semantics of FOL

```

theory FOLPF: LF {
  include FOL
  forallIntro :  $\Pi F:term \rightarrow prop.$ 
                 $(\Pi x:term. \vdash (F x)) \rightarrow \vdash \forall (\lambda x:term. F x)$ 
  forallElim  :  $\Pi F:term \rightarrow prop.$ 
                 $\vdash \forall (\lambda x:term. F x) \rightarrow \Pi x:term. \vdash (F x)$ 
}

```

rules are constants

Small Scale Example (3)

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
  ...
}

```

Abstract Syntax of Terms

Key ideas

- ▶ no predefined constants
- ▶ single general syntax tree constructor $c(\Gamma; \vec{E})$
- ▶ $c(\Gamma; \vec{E})$ binds variables and takes arguments
 - ▶ non-binding operators: Γ empty e.g., `apply(\cdot ; f , a)` for $(f\ a)$
 - ▶ typical binders: Γ and \vec{E} have length 1
e.g., `lambda($x:A$; t)` for $\lambda x:A.t$

contexts	Γ	$::=$	$(x[: E][= E])^*$
terms	E	$::=$	
constants			c
variables			x
complex terms			$c(\Gamma; E^*)$

Terms are relative to theory T that declares the constants c

Concrete Syntax of Terms

- ▶ Theories may attach notation(s) to each constant declaration
- ▶ Notations of c introduce concrete syntax for $c(\Gamma; \vec{E})$

e.g., for type theory

concrete syntax	constant declaration	abstract syntax
$E ::=$		
type	type #	type
$\Pi x : E_1. E_2$	Pi # $\Pi V1 . 2$	Pi($x : E_1; E_2$)
$E_1 \rightarrow E_2$	arrow # $1 \rightarrow 2$	arrow($\cdot; E_1, E_2$)
$\lambda x : E_1. E_2$	lambda # $\lambda V1 . 2$	lambda($x : E_1; E_2$)
$E_1 E_2$	apply # $1 2$	apply($\cdot; E_1, E_2$)

Judgments

- ▶ MMT terms subsume terms of specific languages
- ▶ Type systems singles out the well-typed terms

For any theory Σ :

$\vdash \Sigma$	$\mathcal{T} = \{\Sigma\}$ is a valid theory definition
$\vdash_{\mathcal{T}} \Gamma$	Γ is a valid context
$\Gamma \vdash_{\mathcal{T}} t : A$	t has type A
$\Gamma \vdash_{\mathcal{T}} E = E'$	E and E' are equal
$\Gamma \vdash_{\mathcal{T}} _ : A$	A is inhabitable

Two kinds of rules:

- ▶ MMT defines some global rules once and for all
foundation-independent rules
- ▶ declared in MMT theories, subject to scoping
foundation-specific rules

Foundation-Independent Rules

- ▶ Lookup rules for atomic terms over a theory $T = \{\Sigma\}$

$$\frac{c : A \text{ in } \Sigma}{\vdash_T c : A} \qquad \frac{c = t \text{ in } \Sigma}{\vdash_T c = t}$$

- ▶ Equivalence and congruence rules for equality
- ▶ Rules for well-formed theories/contexts

$$\frac{\overline{\vdash \cdot} \quad \vdash \Sigma \quad [\vdash_{\Sigma} - : A] \quad [\vdash_T t : A]}{\vdash \Sigma, c[: A][= t]}$$

Foundation-Specific Rules

- ▶ Declared in theories as constants
- ▶ Carry reference to self-contained Scala object implementing a rule interface

~ 10 rule interfaces, in particular one for each algorithm:

- ▶ simplification: $\Gamma \vdash_{\mathcal{T}} E = ?$
- ▶ equality checking: $\Gamma \vdash_{\mathcal{T}} E = E' ?$
- ▶ type inference: $\Gamma \vdash_{\mathcal{T}} t : ?$
- ▶ type checking: $\Gamma \vdash_{\mathcal{T}} t : A ?$
- ▶ proving: $\Gamma \vdash_{\mathcal{T}} ? : A$

experimental

E.g., λ -inference rule

- ▶ applicable to $\Gamma \vdash_{\mathcal{T}} t : ?$ whenever $t = \lambda x : A. s$
- ▶ recursively infers type of s , returns Π -type
- ▶ reports errors and trace messages

MMT Tool

Mature implementation

- ▶ API for representation language foundation-independent
- ▶ Collection of reusable algorithms
no commitment to particular application
- ▶ Extensible wherever reasonable
storage backends, file formats, user interfaces, ...
operators and rules, language features, checkers, ...

Separation of concerns between

- ▶ Foundation developers e.g., language primitives, rules
- ▶ Service developers e.g., search, theorem prover
- ▶ Application developers e.g., IDE, proof assistant

Yields rapid prototyping for logic systems

Logical Results

- ▶ Module system
modularity transparent to foundation developer
- ▶ Concrete/abstract syntax
notation-based parsing/presentation
- ▶ Interpreted symbols, literals
external model/implementation reflected into MMT
- ▶ Type reconstruction
foundation plugin supplies only core rules
- ▶ Simplification
rule-based, integrated with type reconstruction
- ▶ Theorem proving?
- ▶ Code generation? Computation?

Knowledge Management Results

- ▶ Change management recheck only if affected
- ▶ Project management indexing, building
- ▶ Extensible export infrastructure
Scala, SVG graphs, LaTeX, HTML, ...
- ▶ Search, querying substitution-tree and relational index
- ▶ Browser interactive web browser, 2D/3D theory graph viewer
- ▶ Editing IDE-like graphical interface, LaTeX integration

Type Reconstruction

Algorithm

- ▶ MMT implements foundation-independent rules
- ▶ visible foundation-specific rules collected from current context
- ▶ algorithm delegates to foundation-specific rules as needed

General algorithm takes care of

- ▶ unknown meta-variables
- ▶ delaying constraints
- ▶ definition expansion
- ▶ module system

transparent to foundation-specific rules

Case Studies

Rewriting

Add rewriting to any language already defined in MMT

Inspiration: LF modulo (e.g., Dedukti)

- ▶ Key idea: annotate
 - ▶ a binary judgment as a rewrite predicate
 - ▶ axioms for that judgment as rewrite rules
- ▶ Implement MMT plugin that dynamically generates new simplification rules for each annotated axioms
 - ▶ implemented via Change Listener interface
 - ▶ Termination of rewrite system remain user's responsibility
- ▶ Needed work: a few 100 hundred loc for the rule generator

External Side Conditions

Monadic type operator for calling external side conditions

LLF _{\mathcal{P}} paper by Honsell, Liquori, Maksimovic, Scagnetto

- ▶ Special expressions that represent keys
 - ▶ normal MMT expressions
 - ▶ but not part of the type system
- ▶ New rules for keys that run external side condition
- ▶ Variables typed by keys indicate which locks can be opened
 - ▶ declared whenever traversing into the monad
 - ▶ automatically ignored by all other typing rules
- ▶ Add new rules for lock types that look for keys in the context
 - ▶ if found, monad can be inspected
 - ▶ otherwise, discharge external side condition by calling rule for key
- ▶ Needed work: ~ 100 loc for 7 rules

One evening at Dagstuhl with Ivan

Linear Logic

Inspiration: resource semantics to represent linear logic in LF-like structural framework **MMT context lookup obeys structural rules**

- ▶ Already done in LF
 - ▶ Kripke style model, worlds represent available resources
 - ▶ monoid of worlds to represent empty world, union
 - ▶ additional laws represent structural rules
 - e.g., **commutativity for exchange**
- ▶ Problem in LF: requires explicit reasoning in the monoid
 - awkward, inadequate**
- ▶ Solution: add new rules for equality reasoning
- ▶ Needed work: depends on desired reasoning strength, < 100 loc for simple version

Conclusion

Summary

- ▶ MMT: foundation-independent framework for declarative languages
 - ▶ representation language
 - ▶ implementation
- ▶ Easy to instantiate with specific foundations
 - ▶ **rapid prototyping logic systems**
- ▶ Deep foundation-independent results
 - ▶ logical: parsing, type reconstruction, module system, ...
 - ▶ knowledge management: search, browsers, IDE, ...
- ▶ Serious contender for
 - ▶ experimenting with new system ideas
 - ▶ generic applications/services
 - ▶ universal library
 - ▶ system integration platform

MMT-Based Foundation-Independent Results

IDE

- ▶ Inspired by programming language IDEs
- ▶ Components
 - ▶ jEdit text editor (in Java): graphical interface
 - ▶ MMT API (in Scala)
 - ▶ jEdit plugin to tie them together

only ~ 1000 lines of glue code
- ▶ Features
 - ▶ outline view
 - ▶ error list
 - ▶ display of inferred information
 - ▶ type inference of subterms
 - ▶ hyperlinks: jump to definition
 - ▶ search interface
 - ▶ context-sensitive auto-completion: show identifiers that

IDE: Example View

The screenshot shows the jEdit IDE with a file named `pl.mmt` open. The main editor displays the following MMT code:

```

1 namespace http://cds.omdoc.org/examples
2 theory PL : http://cds.omdoc.org/urtheories?LF =
3   prop : type
4   ded  : prop → type
5   and  : prop → prop → prop
6   impl : prop → prop → prop
7   equiv : prop → prop → prop
8   = [x, y] (x ⇒ y) ∧ ded

```

The code is annotated with various symbols: `SS` (blue dashed box), `RS` (green dashed box), and `US` (blue dashed box). The `ded` definition is highlighted in blue.

The sidebar on the left shows a tree view of the file structure:

- pl.mmt
 - theory PL
 - prop
 - ded
 - and
 - impl
 - equiv
 - type
 - definition
 - lambda
 - x
 - prop
 - y
 - prop
 - and
 - impl
 - y
 - ded

The status bar at the bottom indicates "1 error, 0 warnings". The error list shows:

```

C:\other\oaff\test\source\examples\pl.mmt (1 error, 0 warnings)
8: invalid object: http://cds.omdoc.org/examples?PL?equiv?definition: ded
   argument must have domain type
   http://cds.omdoc.org/examples?PL; x:prop, y:prop |- ded : prop
   http://cds.omdoc.org/examples?PL; x:prop, y:prop |- prop→type = prop

```

The bottom status bar shows the coordinates `8,30` and the text `(mmt,sidekick,UTF-8)Smr oWV 27/3 Mb 4 error(s)19:50`.

An Interactive Library Browser

- ▶ MMT content presented as HTML5+MathML pages
- ▶ Dynamic page updates via Ajax
- ▶ MMT used through HTTP interface with JavaScript wrapper
- ▶ Features
 - ▶ interactive display e.g., inferred types, redundant brackets
 - ▶ smart navigation via MMT ontology
 - ▶ can be synchronized with jEdit
 - ▶ dynamic computation of content
 - ▶ e.g., definition lookup, type inference
 - ▶ graph view: theory diagram as SVG

Browser: Example View

The MMT Web Server

[Graph View](#)
[Search](#)
[Administration](#)
[Help](#)

code.google.com / p / hol-light / source / browse / trunk ? bool

Style: html5

- hollight
- ├ arith.omdoc
- ├ bool.omdoc
- ├ calc_int.omdoc
- ├ calc_num.omdoc
- ├ calc_rat.omdoc
- ├ cart.omdoc
- ├ class.omdoc
- ├ define.omdoc
- ├ ind_defs.omdoc
- ├ ind_types.omdoc
- ├ int.omdoc
- ├ iterate.omdoc
- ├ lists.omdoc
- ├ nums.omdoc
- ├ pair.omdoc
- ├ real.omdoc
- ├ realarith.omdoc
- ├ relax.omdoc
- ├ sets.omdoc

bool

T [show/hide type](#) [show/hide onedim-notation](#) [show/hide tags](#) [show/hide metadata](#)

T_DEF [show/hide type](#) [show/hide tags](#) [show/hide metadata](#)

TRUTH [show/hide type](#) [show/hide definition](#) [show/hide tags](#) [show/hide metadata](#)

^ [show/hide type](#) [show/hide onedim-notation](#) [show/hide tags](#) [show/hide metadata](#)

AND_DEF [show/hide type](#) [show/hide tags](#) [show/hide metadata](#)

=> [show/hide type](#) [show/hide onedim-notation](#) [show/hide tags](#) [show/hide metadata](#)

IMP_DEF [show/hide type](#) [show/hide tags](#) [show/hide metadata](#)

! [show/hide type](#) [show/hide onedim-notation](#) [show/hide tags](#) [show/hide metadata](#)

type $\{A : \text{holtype}\} (A \Rightarrow \text{bool}) \Rightarrow \text{bool}$

onedim-notation $\forall x:_. a$ (precedence 0)

FORALL_DEF [show/hide type](#) [show/hide tags](#) [show/hide metadata](#)

type $\{A : \text{holtype}\} \vdash (! A) = \lambda P:A \Rightarrow \text{bool} . P = \lambda x:A . T$

<http://latin.omdoc.org/foundations/hollight?Kernel?fun>

Browser Features: 2-dimensional Notations

REAL_POW_DIV

show/hide type

show/hide definition

type $\vdash \forall x:\text{real} . \forall y:\text{real} . \forall n:\text{num} . \left(\frac{x}{y}\right)^n = \frac{x^n}{y^n}$

Browser Features: Proof Trees

The MMT Web Server
[Graph View](#) [Administration](#) [Help](#)

Style: html5 cds.omdoc.org / courses / 2013 / ACS1 / exercise_10.mmt ? Problem3

acs1_2013

- exercise_10.omdoc
 - Problem2
 - Problem3**
 - Problem4
- example
- latin
- lmfdb
- mathscheme
- mml
- openmath
- test
- tptp
- urtheories

theory Problem3 meta LF

include : <http://cds.omdoc.org/examples?FOLEQNatDed>

circ : term → term → term

e : term

R : ⊢ ∀ x x ∘ e ≐ x

C : ⊢ ∀ x ∀ y x ∘ y ≐ y ∘ x

L : ⊢ ∀ x e ∘ x ≐ x

$$= \left[\begin{array}{c} \frac{\frac{\frac{C\ e}{\vdash \forall y x \circ y \doteq y \circ e} \text{krallE } x}{\vdash e \circ x \doteq x \circ e} \text{krallE} \quad \frac{R\ x}{\vdash x \circ e \doteq x} \text{krallE}}{\vdash e \circ x \doteq x} \end{array} \right]$$

⊢ ∀ x e ∘ x ≐ x

Enter an object over theory: <http://cds.omdoc.org/courses/2013>

[x] x ∘ e

analyze simplify

[x] x ∘ e

[x:term] term

- reconstructed types >
- implicit arguments >
- redundant brackets >**
- infer type >
- simplify >
- fold >

show

hide

Browser Features: Type Inference

The screenshot shows a theorem prover interface with several theorem entries. A context menu is open over the first entry, listing various actions. The 'infer type' option is highlighted by the mouse cursor. A secondary window titled 'type' displays the inferred type for the selected theorem.

FORALL_DEF show/hide type show/hide tags show/hide metadata
type $\{A:\text{holtype}\} \vdash (!A) = \lambda P:A \Rightarrow \text{bool}. P = \lambda x:A.T$

? show/hide type
reconstructed types >

EXISTS_DEF show
implicit arguments >

✓ show/hide type
redundant brackets >

OR_DEF show/hide
infer type (highlighted)

simplify

F show/hide type
fold

type
($A \Rightarrow \text{bool}$) \Rightarrow bool

Browser Features: Parsing

Enter an object over theory:

analyze simplify

result: $[x] \forall y. \exists z. y =_{\text{num}} x + z$

inferred type: $\{x: \text{num}\} \text{bool}$

Example Service: Search

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

Namespace

Theory

Name

Enter an expression over theory

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

Search

type of **MOD_EQ**

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

type of **MOD_MULT_ADD**

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

Example Service: Theory Graph Viewer

Theory graphs with 1000s of nodes

→ special visualization tools needed

recently even in 3D



demo at <https://www.youtube.com/watch?v=Mx7HSWD5dwg>

L^AT_EX Integration

- ▶ upper part: L^AT_EX source for the item on associativity
- ▶ lower part: pdf after compiling with L^AT_EX-MMT
- ▶ enriched with type inference, cross references, tooltips
e.g., type argument M of equality symbol

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