

Representing Logics and Logic Translations

Florian Rabe

Jacobs University Bremen, Carnegie Mellon University

19.06.2008

Introduction

Premise

Set/model theory

Type/proof theory

Logic translations

Knowledge management

Premise

Institutions

Set/model theory

LF

Type/proof theory

Logic translations

Knowledge management

OMDoc

What is Logic?

- ▶ Field of mathematics concerned with propositions and their truth
- ▶ Concepts of *proposition* and *truth*
 - ▶ very easy to understand intuitively
 - ▶ at the core of mathematics
 - ▶ very hard to capture formally
- ▶ Seemingly simple issues can raise profound philosophical questions, e.g.,
 - ▶ What does it mean to *define*?
 - ▶ What is $\{x \mid P(x)\}$?

What is a Logic?

- ▶ A logic is a specific formalization of propositions and truth
- ▶ Two uses of logics:
 - ▶ as a foundation of mathematics
 - ▶ as a specification and interface language in mathematics and computer science
- ▶ No best formalization, a multitude of logics in use

Representing Logics and Logic Translations

- ▶ Logical framework: a definition of what a logic is
- ▶ Logic translations only within a logical framework
- ▶ Representation: definition or encoding of a logic (translation) in a framework
- ▶ Main question: What should the logical framework be?

A Brief History of Logic

1879	Frege's Begriffsschrift, the first formal logic
1883	Cantor's Grundlagen, (naive) set theory
1889	Peano, axiomatization of the natural numbers
1901	Russell, paradoxon in set theory and Frege's system
1908, 1913	Russell, Whitehead, the first type theory
1908, 1922	Zermelo, Fraenkel, the first axiomatic set theory
1920s	Hilbert, reduction of truth to effective means
1929, 1936	Gödel, Gentzen, proof theory of first-order logic
1931	Gödel, unprovability of consistency
1933, 1956	Tarski, principles of model theory
1940	Church, simple type theory
1942	Eilenberg, Mac Lane, category theory
1958, 1969	Curry-Howard correspondence, principles of proof theory
1974	Martin-Löf, dependent type theory
1978	Goguen, Burstall, institutions
1993	Harper, Honsell, Plotkin, LF

Model Theory

- ▶ Propositions contain meaningless symbols, e.g., f in $\forall x, y. f(x, y) = f(y, x)$
- ▶ Models assign meanings to symbols, e.g., interpret f as addition or subtraction of integers
- ▶ Satisfaction: a proposition is *true* or *false* in a model
- ▶ Consequence:

$$A_1, \dots, A_n \models B$$

“ B is a consequence of the A_i iff B is true in all models in which the A_i are true.”

Proof Theory

- ▶ Judgments express properties of objects, e.g., *true A*
- ▶ Rules use judgments as hypotheses and conclusions, e.g.,

$$\frac{\textit{true A} \quad \textit{true B}}{\textit{true (A \wedge B)}}$$

- ▶ Proofs are trees with judgments as nodes that are connected by rules, e.g.,

$$\frac{\frac{\textit{true A} \quad \textit{true B}}{\textit{true (A \wedge B)}} \quad \textit{true C}}{\textit{true (A \wedge B \wedge C)}}$$

- ▶ Consequence:

$$A_1, \dots, A_n \vdash B$$

“*B* is a consequence of the *A_i* if there is a proof tree with root *true B* and leaves *true A_i*.”

Motivation (1)

- ▶ Model and proof theory complementary (soundness, completeness), but also in philosophical conflict
- ▶ Successful logical frameworks focusing on one of the two:
 - ▶ institutions, the most successful model theoretical framework
 - ▶ dependent type theory (LF), the most successful proof theoretical framework

Motivation (1)

- ▶ Model and proof theory complementary (soundness, completeness), but also in philosophical conflict
- ▶ Successful logical frameworks focusing on one of the two:
 - ▶ institutions, the most successful model theoretical framework
 - ▶ dependent type theory (LF), the most successful proof theoretical framework
- ▶ Complementary strengths:
 - ▶ institutions: abstraction over concrete syntax, strong support for translations
 - ▶ dependent type theory: elegant syntax, strong support for effective reasoning
- ▶ Goal: Give a logical framework that combines and subsumes institutions and dependent type theory

Logical Knowledge Management (LKM)

- ▶ Sophisticated LKM services needed for logic on a large scale
 - ▶ module system: distributed development, reuse, encapsulation
 - ▶ storage: efficient retrieval, intelligent search, versioning
 - ▶ display to the user in interactive web-based environments
 - ▶ management of change: dependency analysis, difference-based infrastructure
- ▶ No comprehensive LKM support in logics and logical frameworks

Logical Knowledge Management (LKM)

- ▶ Sophisticated LKM services needed for logic on a large scale
 - ▶ module system: distributed development, reuse, encapsulation
 - ▶ storage: efficient retrieval, intelligent search, versioning
 - ▶ display to the user in interactive web-based environments
 - ▶ management of change: dependency analysis, difference-based infrastructure
- ▶ No comprehensive LKM support in logics and logical frameworks
- ▶ LKM services often independent of the underlying logic, generic implementations possible
- ▶ No satisfactory interface languages between logical systems and LKM services
- ▶ Most advanced language: OMDoc

Motivation (2)

- ▶ Give a generic scalable representation language for logical knowledge
- ▶ Provide an interface between logical systems and LKM services
- ▶ Crucial difficulties:
 - ▶ trade-off between simplicity and expressivity
 - ▶ trade-off between formal semantics and syntactical freedom
 - ▶ appeal to researchers in all fields — no foundational commitment

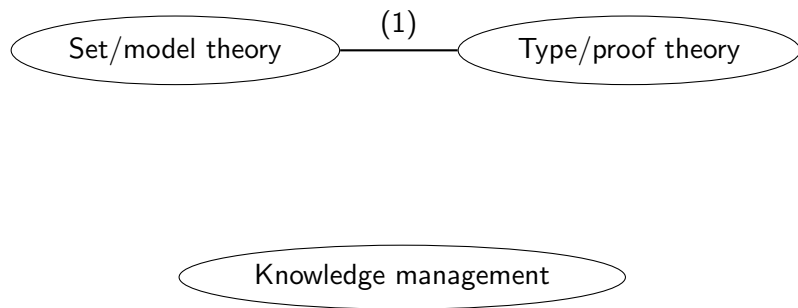
Overview

Set/model theory

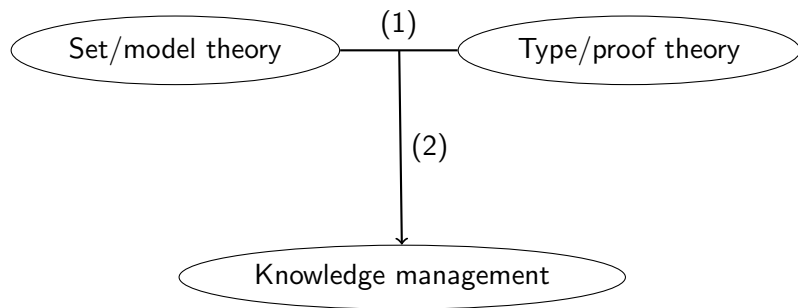
Type/proof theory

Knowledge management

Overview



Overview



Part 1: Combining Model and Proof Theory

Overview

1. Add abstraction over proof theory to institutions
yields definition of *a logic*
2. Model theory for dependent type theory (with Steve Awodey)
3. Mark up otherwise implicit information
yields a logic \mathbb{L} for dependent type theory
4. Represent logics \mathbb{I} via logic comorphisms from \mathbb{I} to \mathbb{L}
5. Represent logic comorphisms from \mathbb{I}_1 to \mathbb{I}_2 as logic comorphism modifications

1. Institutions with Proofs

$\sigma : \Sigma \rightarrow \Sigma'$ Sig-morphism in an institution (Sig, Sen, Mod, \models)
e.g., $\Sigma = Monoid$, $\Sigma' = Group$, σ inclusion

$$\begin{array}{ccc} |Mod(\Sigma)| & \xleftarrow{|Mod(\sigma)|} & |Mod(\Sigma')| \\ \models_{\Sigma} \Big| & & \Big| \models_{\Sigma'} \\ Sen(\Sigma) & \xrightarrow{Sen(\sigma)} & Sen(\Sigma') \end{array}$$

1. Institutions with Proofs

$\sigma : \Sigma \rightarrow \Sigma'$ *Sig*-morphism in an institution $(\text{Sig}, \text{Sen}, \text{Mod}, \models)$
e.g., $\Sigma = \text{Monoid}$, $\Sigma' = \text{Group}$, σ inclusion

$$\begin{array}{ccc} |\text{Mod}(\Sigma)| & \xleftarrow{|\text{Mod}(\sigma)|} & |\text{Mod}(\Sigma')| \\ \downarrow \models_{\Sigma} & & \downarrow \models_{\Sigma'} \\ \text{Sen}(\Sigma) & \xrightarrow{\text{Sen}(\sigma)} & \text{Sen}(\Sigma') \\ \downarrow \text{true}_{\Sigma} & & \downarrow \text{true}_{\Sigma'} \\ |\text{Pf}(\Sigma)| & \xrightarrow{|\text{Pf}(\sigma)|} & |\text{Pf}(\Sigma')| \end{array}$$

1. Institutions with Proofs, formally

Logic ($Sig, Sen, Mod, \models, Pf, true$)

- ▶ Syntax

- ▶ Sig category of signatures
- ▶ $Sen : Sig \rightarrow \mathcal{SET}$: functor assigning to signatures sets of sentences

- ▶ Model theory

- ▶ $Mod : Sig \rightarrow \mathcal{CAT}^{op}$: functor assigning to signatures categories of models
- ▶ $\models : Sen \rightarrow |-|^r \circ Mod$ natural transformation assigning to every signature Σ the satisfaction relation $\models_{\Sigma} \subseteq Sen(\Sigma) \times |Mod(\Sigma)|$ between the sets and the models

$|-|^r$: forgetful functor from \mathcal{CAT} to classes and relations

1. Institutions with Proofs, formally

Logic ($Sig, Sen, Mod, \models, Pf, true$)

- ▶ Syntax

- ▶ Sig category of signatures
- ▶ $Sen : Sig \rightarrow \mathcal{SET}$: functor assigning to signatures sets of sentences

- ▶ Model theory

- ▶ $Mod : Sig \rightarrow \mathcal{CAT}^{op}$: functor assigning to signatures categories of models
- ▶ $\models : Sen \rightarrow |-|^r \circ Mod$ natural transformation assigning to every signature Σ the satisfaction relation $\models_{\Sigma} \subseteq Sen(\Sigma) \times |Mod(\Sigma)|$ between the sets and the models

- ▶ Proof theory

- ▶ $Pf : Sig \rightarrow \mathcal{PFCAT}$: functor assigning to signatures categories — objects are judgments, morphisms are proofs
- ▶ $true : Sen \rightarrow |-|^r \circ Pf$ natural transformation assigning to every signature Σ a mapping $true_{\Sigma} : Sen(\Sigma) \rightarrow |Pf(\Sigma)|$

$|-|^r$: forgetful functor from \mathcal{CAT} to classes and relations

\mathcal{PFCAT} : Category of categories with products

2. Model theory for dependent type theory

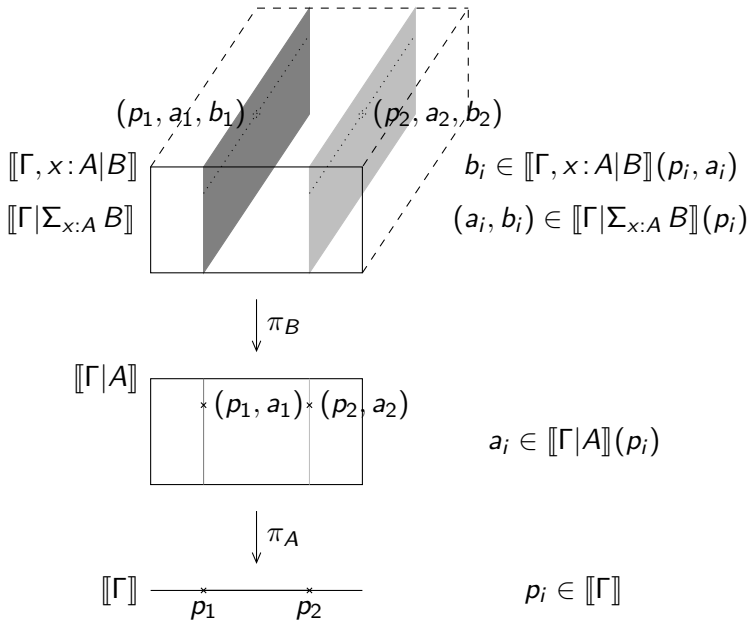
- ▶ Dependent type theory with type-valued functions, sums, products, identity types, unit type, and empty type
- ▶ Existing model theories rather abstract based on locally cartesian closed categories

2. Model theory for dependent type theory

- ▶ Dependent type theory with type-valued functions, sums, products, identity types, unit type, and empty type
- ▶ Existing model theories rather abstract based on locally cartesian closed categories
- ▶ General idea: use Kripke models, interpret types as sets indexed by a poset of worlds

2. Model theory for dependent type theory

- ▶ Dependent type theory with type-valued functions, sums, products, identity types, unit type, and empty type
- ▶ Existing model theories rather abstract based on locally cartesian closed categories
- ▶ General idea: use Kripke models, interpret types as sets indexed by a poset of worlds
- ▶ Trickiest question: Soundness
 - proofs of well-definedness, substitution property, and soundness by mutually recursive inductions over well-formedness derivations
- ▶ Completeness result (new also for the simply-typed case)



3. The Logic \mathbb{L} , syntax and models

- ▶ Signatures of the logic \mathbb{L}
 - ▶ signatures of dependent type theory
 - ▶ role assignments to distinguish sorts and judgments, and constants and proof rules
 - ▶ a distinguished sort \underline{o} : type of formulas
 - ▶ a distinguished judgment $\underline{true}: \underline{o} \rightarrow \text{type}$ for the truth judgment
- ▶ \mathbb{L} -models interpret judgments as singletons (true) or empty sets (false)
- ▶ Proof irrelevance

3. The logic \mathbb{L} , proof categories

- ▶ Objects of the proof category $Pf^{\mathbb{L}}(\Sigma)$: contexts

$$\dots, x_i : J_i, \dots$$

for some countable set of I indices and judgments J_i

- ▶ Morphisms in $Pf^{\mathbb{L}}(\Sigma)$

$$\text{from } \Gamma \text{ to } \dots, x_i : J_i, \dots$$

are substitutions, i.e., families

$$(p_i)_{i \in I} \quad \text{where} \quad \Gamma \vdash_{\Sigma} p_i : J_i$$

4. Representation example: first-order logic

```
% sorts, constants
i : type.
o : type.
tt : o.
ff : o.
imp : o -> o -> o.
forall : (i -> o) -> o.

% judgments, rules
true : o -> type.
ttl : true top.
ffE : true bot -> true A.
impl : (true A -> true B) -> true (A imp B).
implE : true (A imp B) -> true A -> true B.
forallI : ({x} true (A x)) -> true (forall A).
forallE : true (forall A) -> {x} true (A x).

% axioms for model theory
ne : i. % non-empty universe
cons : true ff -> empty. % consistency
```

4., 5. Representing Logics and Translations in \mathbb{L}

- ▶ \mathbb{L} meta-logic to define and encode logics and translations
- ▶ Logic definition: signature category Sig together with functor $\Phi : Sig \rightarrow Sig^{\mathbb{L}}$ induces logic $(Sig, Sen^{\mathbb{L}} \circ \Phi, Mod^{\mathbb{L}} \circ \Phi, \models_{\Phi}^{\mathbb{L}}, Pf^{\mathbb{L}} \circ \Phi, true_{\Phi}^{\mathbb{L}})$

4., 5. Representing Logics and Translations in \mathbb{L}

- ▶ \mathbb{L} meta-logic to define and encode logics and translations
- ▶ Logic definition: signature category Sig together with functor $\Phi : Sig \rightarrow Sig^{\mathbb{L}}$ induces logic $(Sig, Sen^{\mathbb{L}} \circ \Phi, Mod^{\mathbb{L}} \circ \Phi, \models_{\Phi}^{\mathbb{L}}, Pf^{\mathbb{L}} \circ \Phi, true_{\Phi}^{\mathbb{L}})$
- ▶ Logic encoding: as logic definition, but with sentence, model, and proof translation between encoded logic and \mathbb{L}
- ▶ Adequacy of logic encodings subsumes model expansion property of institutions and adequacy of LF-encodings

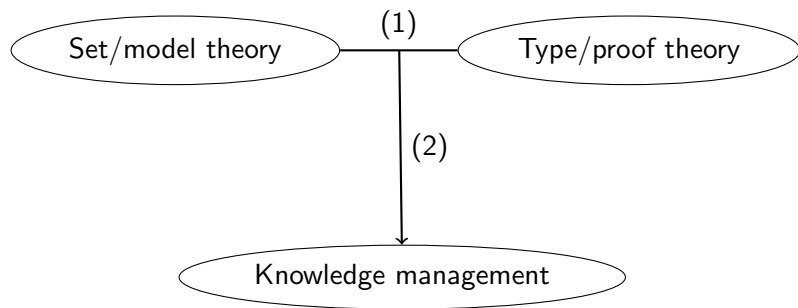
Conclusion

- ▶ \mathbb{L} combines advantages of institutions and LF
- ▶ Existing encodings subsumed
- ▶ Model theoretical view: \mathbb{L} provides specification language for institutions and institution comorphisms
- ▶ Proof theoretical view: \mathbb{L} provides set-theoretic semantics of the meta-logic LF

Conclusion

- ▶ \mathbb{L} combines advantages of institutions and LF
- ▶ Existing encodings subsumed
- ▶ Model theoretical view: \mathbb{L} provides specification language for institutions and institution comorphisms
- ▶ Proof theoretical view: \mathbb{L} provides set-theoretic semantics of the meta-logic LF
- ▶ Future work: Extensions
 - ▶ non-compositional translations
 - ▶ better support for translations of types to subtypes
- ▶ Future work: Applications
 - ▶ integration of logic and translation definitions into Hets
 - ▶ abstract completeness analysis

Part 2: Logical Knowledge Management

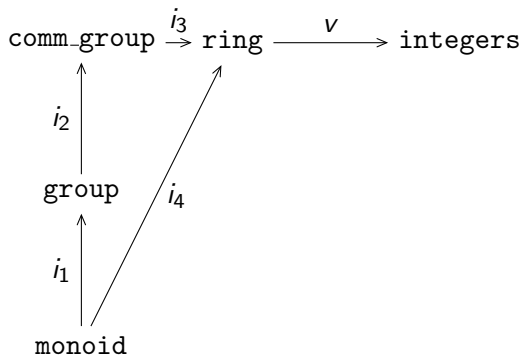


MMT: A Knowledge Representation Language for Logic

- ▶ Four levels of knowledge: objects, statements, theories, documents
- ▶ Module system using theories and theory morphisms as the central primitives
- ▶ Logical theories, logics, and logical frameworks uniformly modelled as theories
- ▶ Abstraction from foundation

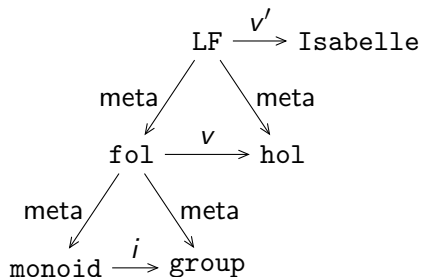
Theory Morphisms

- ▶ Theory graph: theories as nodes, imports and views as edges, morphisms as paths
- ▶ Imports: definitional flavor
- ▶ Views: theorem flavor



Meta-Theories

- ▶ Meta-relation between languages pervades mathematical practice
- ▶ Examples: logical framework to logic, logic to theory, foundation to domain of numbers, programming language to program



Document and Theory Level

Document level

- ▶ Library: list of document declarations
- ▶ Document declaration $g := \{\gamma\}$
 g URI, γ list of theory and view declarations

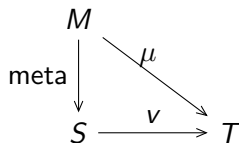
Document and Theory Level

Document level

- ▶ Library: list of document declarations
- ▶ Document declaration $g := \{\gamma\}$
 g URI, γ list of theory and view declarations

Theory level

- ▶ Theory declaration: $S \stackrel{M}{:=} \{\vartheta\}$
 S name, M meta-theory, ϑ symbol declarations
- ▶ View declaration: $v : S \rightarrow T \stackrel{\mu}{:=} \{\sigma\}$
 v name, S domain, T codomain,
 μ meta-morphism,
 σ assignments to symbols



Statement and Object Level

- ▶ Constant c with type τ and definition δ : $c := \delta : \tau$
- ▶ Terms τ, δ : constants, variables, application, binding, attribution, morphism application

atomic object	composed object	type	checked relative to
constant	term	term	home theory

Statement and Object Level

- ▶ Constant c with type τ and definition δ : $c := \delta : \tau$
- ▶ Terms τ, δ : constants, variables, application, binding, attribution, morphism application
- ▶ Import i from theory S with instantiation σ : $i : S := \{\sigma\}$
- ▶ Morphisms: imports, views, identity, composition

atomic object	composed object	type	checked relative to
constant	term	term	home theory
import	morphism	domain	codomain

Intuitions

Intuition of a theory $T := \{\vartheta, i: S := \{\sigma\}\}$:

- ▶ Module system: application functor S with instantiation σ
- ▶ Development graphs: definitional link from S to T

- ▶ Category theory: pushout of

$$\begin{array}{ccc} \text{dom}(\sigma) & \xrightarrow{\sigma} & \vartheta \\ \downarrow & & \\ \text{body}(S) & & \end{array}$$

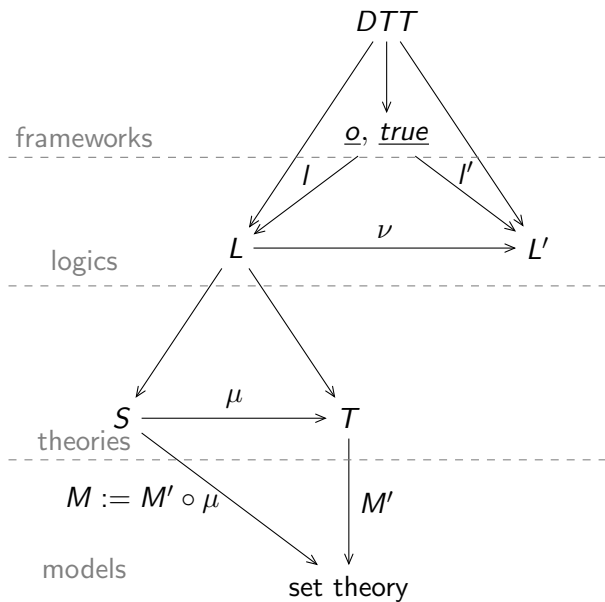
Intuition of a view $v: S \rightarrow T := \{\sigma\}$:

- ▶ Module system: structure v conforms to, implements, interprets S in context T
- ▶ Development graphs: v theorem link from S to T
- ▶ Category theory: σ theory morphism from $\text{body}(S)$ to $\text{body}(T)$

Foundations

- ▶ Well-formedness of MMT-expression defined by inference system
- ▶ Judgments for typing and equality defined by foundation
- ▶ Module system, data maintenance, and state transparent to foundation
- ▶ Examples:
 - ▶ foundation for dependent type theory declares constants for Π , λ , etc.
 - ▶ foundation for set theory declares constants for \mathcal{SET} , \in , etc.; all sets have type \mathcal{SET}

Representing \mathbb{L} in MMT



Scalability

- ▶ Flexible flattening: all module system constructs can be eliminated separately
- ▶ Naming scheme: all declared or imported symbols accessible by globally unique URIs
- ▶ Module system: simple enough to translate out of; expressive enough to translate into
- ▶ XML, OpenMath as concrete syntax, core of upcoming OMDoc format
- ▶ Foundation-independence: implementation of MMT services without knowing the foundation

Future Work: Theory

- ▶ Structured heterogeneous proofs
- ▶ Semi-formal documents
- ▶ More complex translations, e.g., partial, non-compositional translations, reflections
- ▶ Model morphisms
- ▶ Logic atlas: inventory of logics representing their relations as theory morphisms

Future Work: Practice

- ▶ API: Scala bindings for all concepts, validation, foundation implementation as plugins
- ▶ Storage: restful HTTP interface as frontend of MMT-aware database
- ▶ Management of change: incremental validity check of additions, deletions, updates
- ▶ Presentation: combination of notation definitions and stylesheet infrastructure to render MMT expressions for humans and machines

Conclusion

- ▶ MMT simple, scalable, but expressive representation language for logical knowledge
- ▶ Meaningful representations while abstracting from foundations
- ▶ Integration of model and proof theory-based frameworks
- ▶ Abstraction layer on top of logical systems
 - ▶ connects logical systems and LKM services
 - ▶ facilitates translations between logical systems

Simple Things I Cannot Do

- ▶ Kripke models: elementary completeness proof
currently: using involved topos theory
- ▶ \mathbb{L} : representation of Henkin models of higher-order logic
problem: models non-standard
- ▶ \mathbb{L} : translation of sorted to unsorted logics
problem: translation of types to subtypes
- ▶ MMT: translation of LF encoded in LF to LF
problem: conflation of type hierarchy levels
- ▶ MMT: extra-logical declarations, e.g., notations, tactics
problem: very specific to systems
- ▶ MMT implementation: type reconstruction for LF
problem: hard to reuse existing code