A Query Language for Formal Mathematical Libraries

Florian Rabe

Jacobs University Bremen, Germany

A Query Language for Formal Mathematical Libraries

Florian Rabe

Jacobs University Bremen, Germany

Scope here: formalized math
but approach extends to presentation, narrative

Querying as an MKM Application

Natural fit!

- MKM excels for large knowledge bases
- That's where querying is most needed
- Still lots of work to do — e.g., see MIR workshop
- Big problem in my and other people's work

Querying as an MKM Application

Natural fit!

- MKM excels for large knowledge bases
- That's where querying is most needed
- Still lots of work to do e.g., see MIR workshop
- Big problem in my and other people's work

Consider Michael Kohlhase's example query:
It looks like this and there was a talk about it at CICM in 2010.

Motivation: LATIN

- LATIN: an atlas of logic formalizations
  - written in modular LF/Twelf
  - 4 years, $\sim$ 10 authors, $\sim$ 1000 modules
  - systematically modular
  - highly interconnected network of LF theories
- Inherently difficult to keep overview, let alone query
- Even difficult to see
  - which declarations does this symbol s depend on?
  - which theories import theory t?
  - ...

Motivation

- Aspinall, Denney, Lüth, Querying Proofs
  CICM 2011, work in progress; LPAR 2012
- My reaction: they could use my MMT language

Their goals	
Which axioms occur in the proof?	
Which witnesses are used for existentials?	
Which tactic uses this axiom?	
Where does this goal come from?	
Why does this tactic not apply?	
What are the goal inputs to tactic t at some point?	
Show me tactic instances using this axiom?	
Show me proven goals which rely on this axiom?	
Is there a sub-proof that occurs more than once?	
Are there duplicated subproofs in the proof?	
Are there steps in the proof which have no effect?	

Motivation

- Aspinall, Denney, Lüth, Querying Proofs
  CICM 2011, work in progress; LPAR 2012
- My reaction: they could use my MMT language

Their goals	in MMT
Which axioms occur in the proof?	trivial
Which witnesses are used for existentials?	trivial
Which tactic uses this axiom?	trivial
Where does this goal come from?	doable
Why does this tactic not apply?	doable
What are the goal inputs to tactic t at some point?	trivial
Show me tactic instances using this axiom?	trivial
Show me proven goals which rely on this axiom?	trivial
Is there a sub-proof that occurs more than once?	easy
Are there duplicated subproofs in the proof?	easy
Are there steps in the proof which have no effect?	doable

MMT https://trac.kwarc.info/MMT

- Generic declarative language
  theories, morphisms, declarations, expressions
  module system
- OMDoc/OpenMath-based XML syntax with Scala-based API and HTTP server
- Foundation-independent
  - no commitment to particular logic or logical framework
    both represented as MMT theories themselves
  - concise and natural representations of wide variety of systems
    e.g., Twelf, Mizar, TPTP, OWL

MMT-based MKM services

Foundation-independence: MMT services carry over to languages represented in MMT

- presentation — MKM 2008
- interactive browsing — MKM 2009
- database — MKM 2010
- archival, project management — MKM 2011
- change management — Friday, AISC 2012
- editing (work in progress) — tomorrow, UITP 2012

- querying — this talk, MKM 2012

Querying

Querying at Jacobs University

a lot of related work

- Kohlhase et al.: MathWebSearch (e.g., AISC 2012)
  - google-style index of expressions on websites
  - search for websites with expression similar to e
- Zholudev: TNTBase (e.g., Balisage 2009)
  - XML + SVN database of mathematical documents
  - XQuery (programming/query language)
- Lange: RDF, semantic web
  - relational abstraction from data (set of subject-predicate-object triples)
  - SPARQL query language

Querying

Querying at Jacobs University

a lot of related work using very different paradigms

- Kohlhase et al.: MathWebSearch (e.g., AISC 2012)
  - google-style index of expressions on websites
  - search for websites with expression similar to e
- Zholudev: TNTBase (e.g., Balisage 2009)
  - XML + SVN database of mathematical documents
  - XQuery (programming/query language)
- Lange: RDF, semantic web
  - relational abstraction from data (set of subject-predicate-object triples)
  - SPARQL query language

Querying

Querying at Jacobs University

a lot of related work using very different paradigms
that should be integrated

- Kohlhase et al.: MathWebSearch (e.g., AISC 2012)
  - google-style index of expressions on websites
  - search for websites with expression similar to e
- Zholudev: TNTBase (e.g., Balisage 2009)
  - XML + SVN database of mathematical documents
  - XQuery (programming/query language)
- Lange: RDF, semantic web
  - relational abstraction from data (set of subject-predicate-object triples)
  - SPARQL query language

Object queries

- Search for objects similar to query object
  unification, normalization, applicable theorems . . .
- General: MathWebSearch, EgoMath, MIaS, . . .
  good overview in Sojka, Liska, MKM 2011
- Custom variants: e.g., Isabelle, Coq, Matita, Mizar
- Great at what they do
- But: not integrated with other query paradigms, e.g.,
  - find all objects similar to e *that occur in a theorem imported into the current theory*
  - find all constants *whose type* is similar to e

Property queries

- SPARQL: RDF query languages (W3C 2008); conjunctive query answering for description logics
- Custom variants: e.g., Coq, Mizar
- Typical query:

$$SELECT\ x, y, z\ WHERE\ P(x, y) \wedge Q(y, z)$$

  often: P, Q are atomic predicates, especially unary or binary
- fast, easy, straightforward indexing, semantic web support
- Relational data model
  - good for: document structure, theory-import relation, dependency relation
  - bad for: mathematical expressions, transitive closures

Compositional query languages

- XQuery (W3C 2007), ...
- Data model based on XML trees
- Hierarchical queries via XPath
- Complex queries using nested FLWOR expressions

$$\textbf{for } x \textbf{ in } Q \textbf{ let } y = q'(x) \textbf{ where } F(x, y) \textbf{ return } Q''(x, y)$$

- User-defined functions and modules
- Good: strong general purpose language
- Bad:
  - requires XML database for good indexing
  - specializations for mathematics must be integrated into XQuery engine

MKM Querying Solutions

Heavyweight

- XML database with XQuery engine
- integrate math-specific query functions and indices

  TNTBase+MMT: MKM 2010
- integrate relational index and SPARQL queries in XQuery

  done in XSPARQL, 2009

MKM Querying Solutions

Heavyweight

- XML database with XQuery engine
- integrate math-specific query functions and indices

  TNTBase+MMT: MKM 2010
- integrate relational index and SPARQL queries in XQuery

  done in XSPARQL, 2009

Lightweight (this talk)

- MMT-based query language QMT
- simple, expressive, formal semantics, self-contained implementation

MKM Querying Solutions

Heavyweight

- XML database with XQuery engine
- queries run on dedicated server

Lightweight (this talk)

- MMT-based query language QMT
- MMT API: same code can be client or server

MKM Querying Solutions

Heavyweight

- XML database with XQuery engine
- queries run on dedicated server

Lightweight (this talk)

- MMT-based query language QMT
- MMT API: same code can be client or server

Side remark

- Should we assume we are always connected to a server?
- pro: it's the future
- contra: keep it simple
  (Or am I just too old-fashioned here?)

QMT

Atomic expressions	Intended Semantics
base type a	a set of individuals
concept symbol c	a subset of a base type
relation symbol r	a relation between two base types
function symbol f	a typed first-order function
predicate symbol p	a typed first-order predicate

Complex Expressions			
Types	T	$::=$	$a \times \ldots \times a \mid set(a \times \ldots \times a)$
Relations	R	$::=$	$r \mid R^{-1} \mid R^{*} \mid R; R \mid R \cup R \mid R \cap R \mid R \setminus R$
Propositions	F	$::=$	$p(Q, \ldots, Q) \mid \neg F \mid F \wedge F \mid \forall x \in Q.F(x)$
Queries	Q	$::=$	$x \mid f(Q, \ldots, Q) \mid \{Q\}$
		$\mid$	$c \mid R(Q) \mid \bigcup_{x \in Q} Q(x) \mid \{x \in Q \mid F(x)\}$

QMT: Semantics

- Well-typed queries defined by type system
- Compositional denotational semantics
- Safety: well-typed queries have well-defined semantics

Kind of Expression	Denotation
Type T : $\mathtt{type}$	a set
Query Q : T	an element of T
element query Q : T	an element of T
set query Q : $set(T)$	a subset of T
Relation $R < a, a'$	a relation between a and a'
Proposition F : $prop$	a boolean truth value

Querying MMT

Define a QMT signature for MMT

- base types: MMT URIs, OpenMath objects, XML
- concept and relation symbols: MMT ontology
  - concepts: theory, constant, ...
  - relation: declares, includes, uses, depends-on, ...
- function and predicate symbols: methods of MMT API
  - definition lookup
  - type inference
  - subobject access
  - HTML+MathML rendering
  - unification query via MathWebSearch
  - ...

Query Examples

- $R(u)$ returns all v such that $(u, v) \in [\![R]\!]$

  Example: all theories that transitively include the theory u

  $$\mathit{includes}^{*\,-1}(u)$$

- $\{x \in Q | F(x)\}$ returns all $u \in [\![Q]\!]$ such that $[\![F]\!]$ holds at u

  Example: all declarations of theories included into the theory u whose type uses the identifier v

  $$\{x \in (\mathit{includes}^*; \mathit{declares})(u) \,|\, \mathit{occurs}(v, \mathit{type}(x))\}$$

Definable Queries

- Replacement queries: $\{q(x) : x \in Q\}$ defined as $\bigcup_{x \in Q}\{q(x)\}$

Definable Queries

- Replacement queries: $\{q(x) \,:\, x \in Q\}$ defined as $\bigcup_{x \in Q}\{q(x)\}$
- DL-style queries: $\Box^c R.Q$ defined as $\{x \in c \,|\, \forall y \in R(x).y \in Q\}$

Definable Queries

- Replacement queries: $\{q(x) : x \in Q\}$ defined as $\bigcup_{x \in Q} \{q(x)\}$
- DL-style queries: $\Box^c R.Q$ defined as $\{x \in c \mid \forall y \in R(x).y \in Q\}$
- XQuery-style queries:

$$\textbf{for } x \textbf{ in } Q \textbf{ let } y = q'(x) \textbf{ where } F(x, y) \textbf{ return } Q''(x, y)$$

defined as $\bigcup_{z \in P} Q''(z_1, z_2)$ where

$$P := \left\{ z \in \{(x, q'(x)) : x \in Q\} \mid F(z_1, z_2) \right\}$$

Technicality 1

All binders relativized by queries: $x \in Q$

- base types may be infinite e.g., OpenMath objects
- but compositional query evaluation yields finite set $[\![Q]\!]$
- thus easy evaluation of all binding expressions

Types	T	$::=$	$a \times \ldots \times a \mid set(a \times \ldots \times a)$
Relations	R	$::=$	$r \mid R^{-1} \mid R^* \mid R; R \mid R \cup R \mid R \cap R \mid R \setminus R$
Propositions	F	$::=$	$p(Q, \ldots, Q) \mid \neg F \mid F \wedge F \mid \forall x \in Q.F(x)$
Queries	Q	$::=$	$x \mid f(Q, \ldots, Q) \mid \{Q\}$
		$\mid$	$c \mid R(Q) \mid \bigcup_{x \in Q} Q(x) \mid \{x \in Q \mid F(x)\}$

Technicality 2

Why both ontology and first-order symbols?

- concept symbol could be unary predicate symbol
- relation symbol could be binary predicate symbol

Relation symbols r and predicate symbols p used differently!

- $[\![R(Q)]\!]$ needs table $[\![R]\!]$
- $\{x \in Q | F(x)\}$ needs boolean-valued function $[\![F]\!]$

Types	T	$::=$	$a \times \ldots \times a \mid set(a \times \ldots \times a)$
Relations	R	$::=$	$r \mid R^{-1} \mid R^* \mid R;R \mid R \cup R \mid R \cap R \mid R \setminus R$
Propositions	F	$::=$	$p(Q, \ldots, Q) \mid \neg F \mid F \wedge F \mid \forall x \in Q.F(x)$
Queries	Q	$::=$	$x \mid f(Q, \ldots, Q) \mid \{Q\}$
		$\mid$	$c \mid R(Q) \mid \bigcup_{x \in Q} Q(x) \mid \{x \in Q \mid F(x)\}$

Implementation

- Document model and relational index maintained by MMT API
- Object index produced by MMT API and read by MathWebSearch
- Queries evaluated by MMT API (HTTP calls to MathWebSearch)
- XML concrete syntax for queries
- Query interface via HTTP POST

Example

- MMT API serving the LATIN atlas: `http://cds.omdoc.org:8080/:query`
- Query: all theories declared in the LATIN atlas

```
<concept name="theory"/>
```

- Query: all identifiers imported into `http://latin.omdoc.org/foundations/zfc?UniversalQuantifier`

```
<related>
  <individual uri="http://latin.omdoc.org/
      foundations/zfc?UniversalQuantifier"/>
  <sequence>
    <transitive>
      <toobject relation="Includes"/>
    </transitive>
    <toobject relation="Declares"/>
  </sequence>
</related>
```

Example

Queries from Javascript

- Ajax-style: QMT request-response cycle hidden from Javascript programmer
- easy to integrate into web pages

```
var query =
 Qpresent(
  Qtype(
   Qsubobject(
    Qcomponent(Qindividual(currentElem), currentComp),
    currentPos),
   'http://cds.omdoc.org/foundations/lf/lf.omdoc?lf'));

execQuery(query,
 function(result){setTypeDialog(result);}
);
```

Your MMT-Based Query Engine

Preparation

1. Implement an export from your language into MMT's XML syntax
2. Register it with MMT
3. Optionally: also register a function that translates your expressions into OpenMath   useful for unification queries

Execution

1. run MMT to export your project
2. run MMT to index it
3. MMT opens a query server
4. optionally: start MathWebSearch and register it with MMT for unification queries

Conclusion

- QMT: a lightweight MMT-based querying solution
  - type system and denotational semantics
  - compositional
  - supports relational queries
  - supports object queries
- Implementation part of the MMT API
  - easy to set up and run
  - platform-independent by using JVM, XML, HTTP
  - easily applicable to your format – requires only export to MMT
- Future work: Widely applicable by extending the signature
  - presentation markup
  - bibliographical data
  - narrative structure