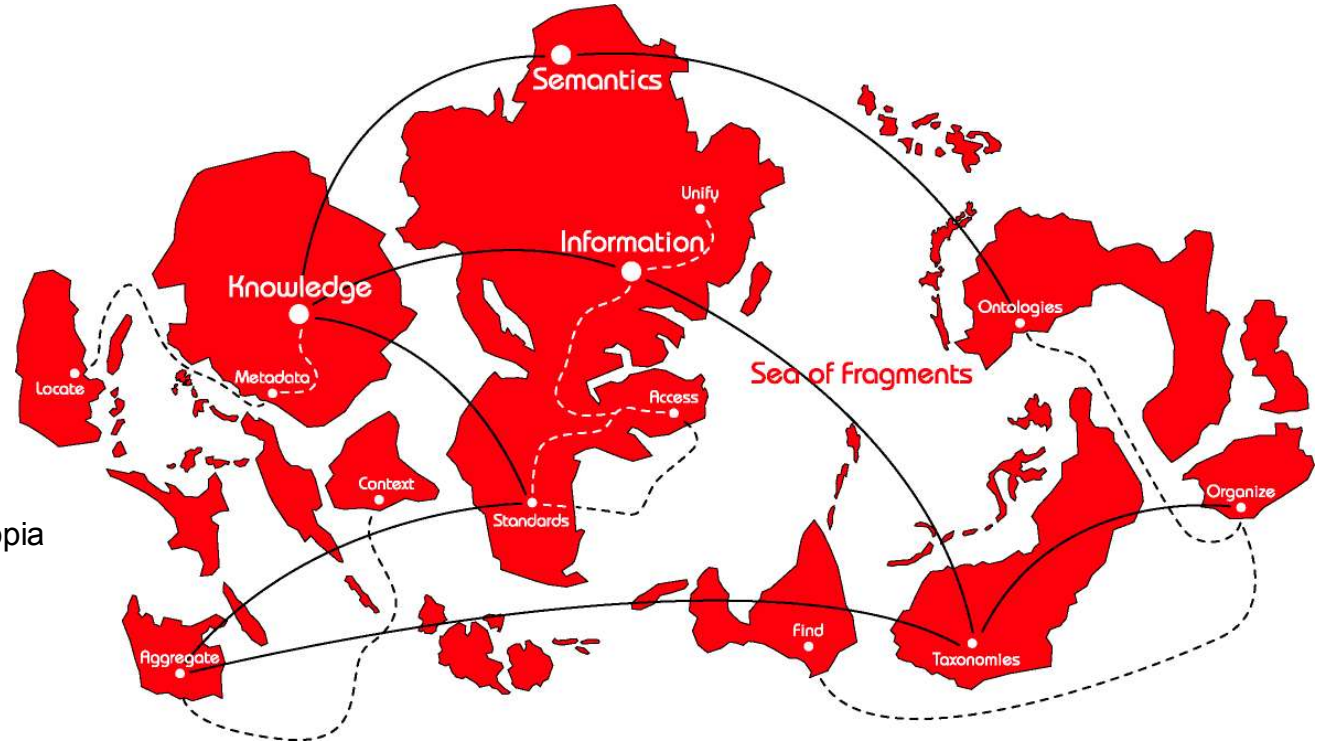
The Q model

A unifying model for RDF and Topic Maps

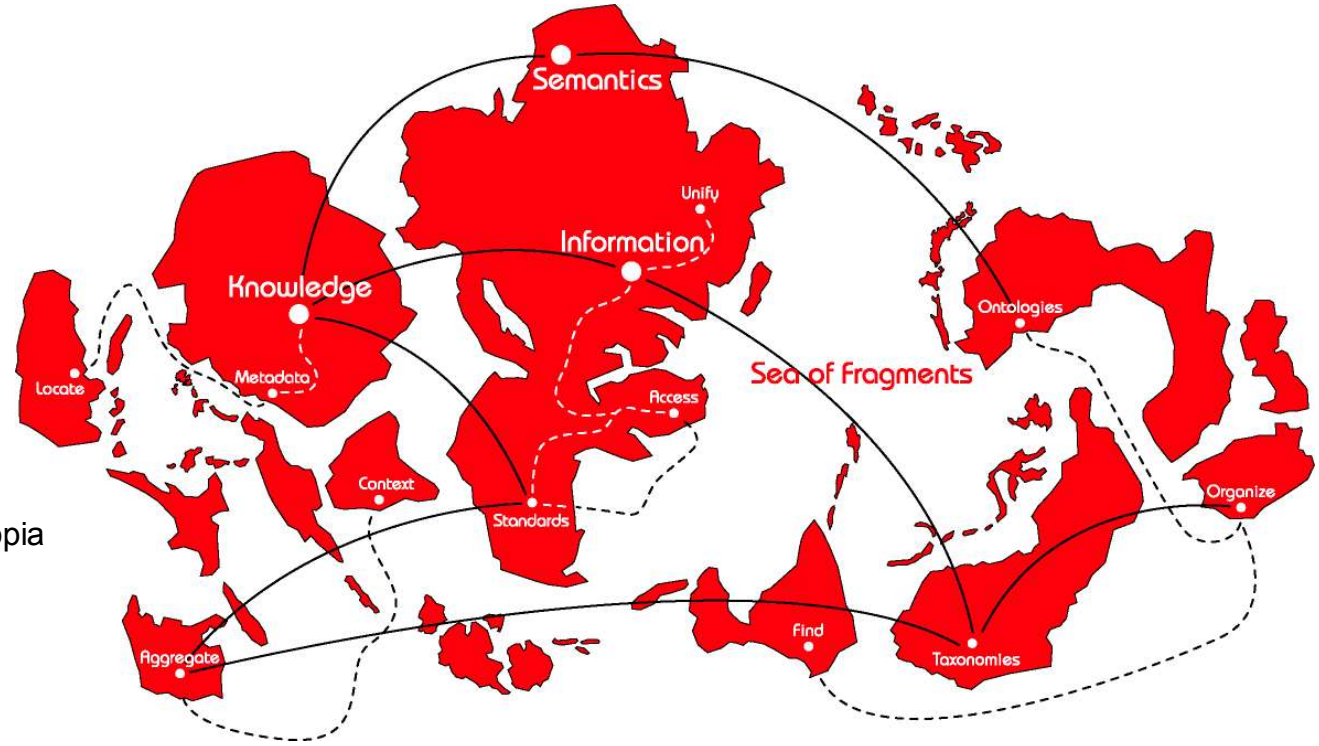


Lars Marius Garshol
Development Manager, Ontopia
<larsga@ontopia.net>

2005-08-02

RDF and Topic Maps

The Saga Continues...



Lars Marius Garshol
Development Manager, Ontopia
<larsga@ontopia.net>

2005-08-02

The goal

- **To create a formal model, Q, such that**
  - any TMDM instance can be transformed into a Q instance with no loss of information
  - any RDF model can be transformed into a Q instance with no loss of information
  - TMDM and RDF have the *same* representation in Q
  - the model must also be efficiently implementable
- **If this were achieved, it would mean that**
  - a single model could serve as the basis for combined RDF/TM implementations
  - also for common RDF/TM query languages
  - OWL/RDFS semantics could be ported to it

Enough already! Why don't you just use RDF instead of this Q thing?

Representing Topic Maps in RDF

And why it's not the way to go

Representing Topic Maps in RDF

- **Topic Maps are higher-level than RDF**
  - that is, Topic Maps have more built-in semantics than RDF triples do
  - to put it another way, RDF is simpler than Topic Maps
- **Therefore, it makes more sense to represent Topic Maps in RDF than the other way around**
  - the direct approach would produce what the W3C survey[1] calls an *object mapping*
- **One attempt at such a representation was made at a nocturne at Knowledge Technologies 2002 in Seattle[2]**
  - later written up and published by yours truly
- **Anne Cregan just presented another attempt at the same**
  - her representation is more up-to-date, and uses OWL
  - however, the same issues apply to both

[1] http://www.w3.org/TR/rdftm-survey/
[2] http://psi.ontopia.net/rdf/

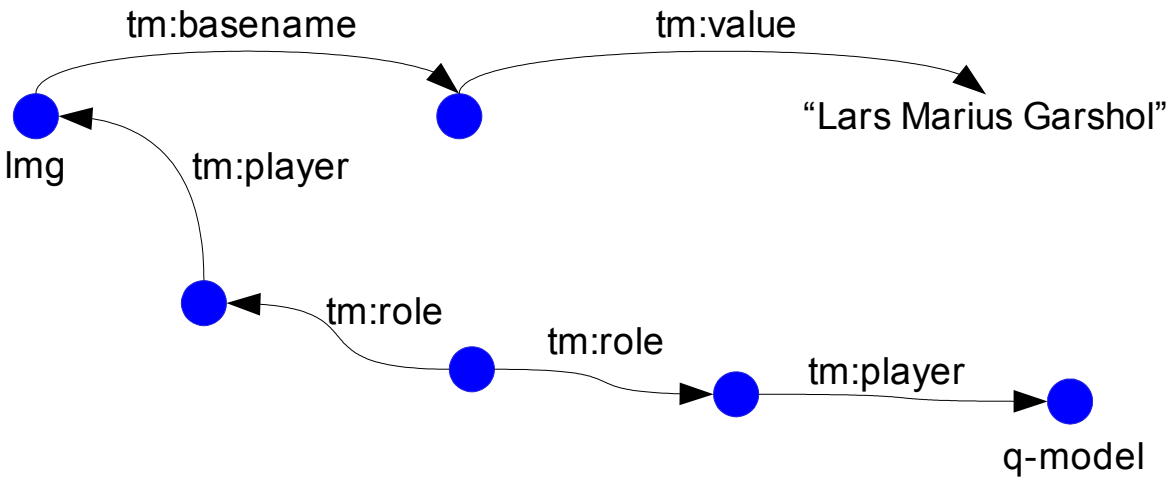
Representing Topic Maps in RDF (2)

- **The following LTM**

  [lmg = “Lars Marius Garshol”]

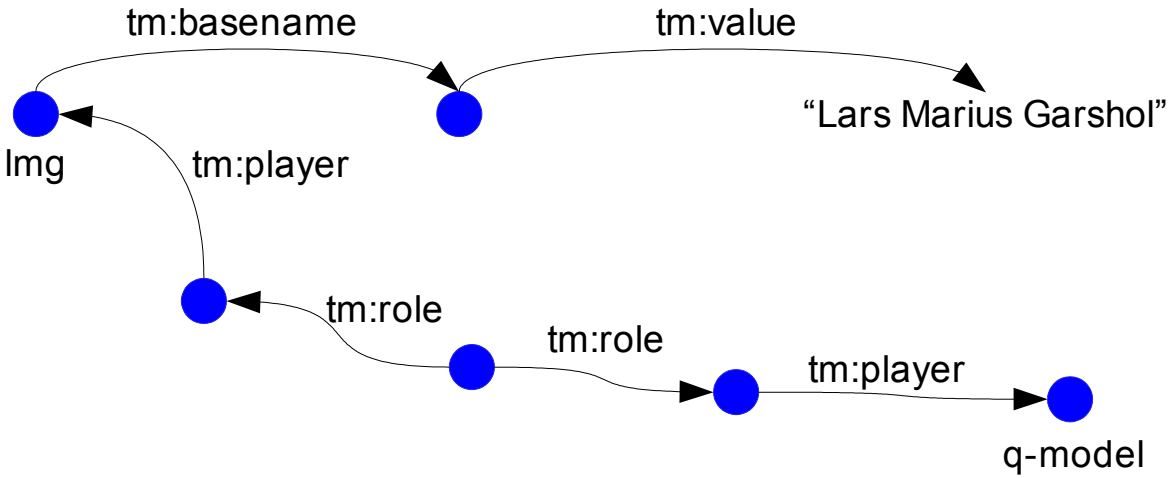
  dc:creator(lmg : dc:value, q-model : dc:resource)

- **would with this approach turn into the following RDF**
  - (some detail omitted)

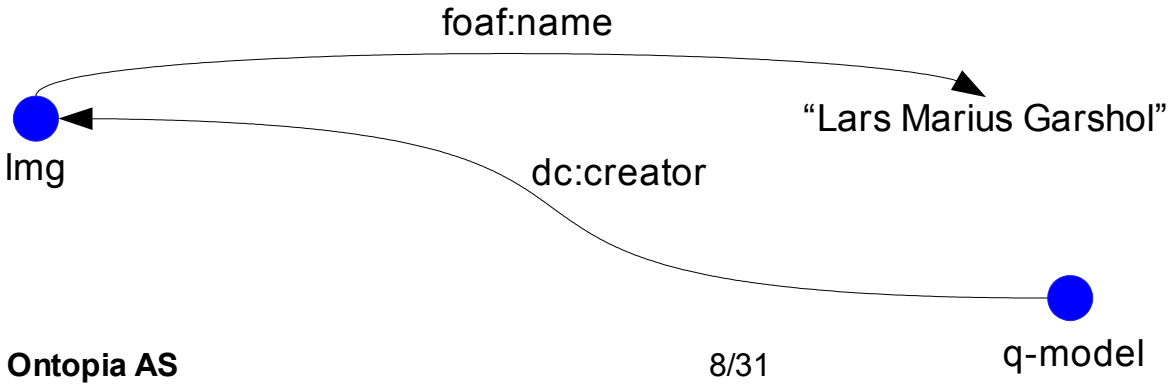


Problem #1: It's not natural

- **This was the RDF we got**



- **This is natural RDF for the same information**



Why do we care about naturalness, anyway?

- **Because if Topic Maps information is represented in this way, then**
  - you can't use RDFS/OWL to constrain your domain vocabulary
  - that is, you can model Topic Maps with RDFS/OWL, but not also your domain
  - it doesn't merge with native RDF information, which uses domain vocabularies
  - queries have to be formulated differently for Topic Maps information and native RDF information on the same subject
  - and so on...
- **In short, an object mapping isn't sufficient for interoperability**
  - you'll always need some form of transformation on the RDF side in order to match up with real RDF data

Problem #2: It's bloated

- **The Italian Opera Topic Map has**
  - 1339 topics, 2411 associations, 1077 occurrences = 4827 TAOs
- **In the 2002 RDF object mapping, this becomes 52673 RDF triples**
  - ie, 11 times the TAO count
  - it can be reduced, but not very substantially
- **For the mondial.xtm topic map you need 288457 RDF triples**
  - that makes it a very big model, but the topic map isn't *that* big
- **Clearly, this is just too voluminous**

So why not make it slim and natural?

- **Why can't we just use a single RDF statement for the base name?**
- **Because in topic maps you also need to represent**
  - the scope of the base name
  - the variant(s) of the base name, if any
  - the topic reifying the base name, if any
- **In RDF there are only two choices:**
  - use reification
    - can get us much closer to natural RDF
    - but causes bloat, since 5 triples are needed for reification
  - use a blank node for the base name (this was the approach taken on slide 7)
- **But, what if there were a way to make reification compact?**

The Q4 model

A naïve approach

The basic idea of Q4

- **We extend the triples of RDF with one more element**
  - that new element represents the identity of the statement
  - (subject, property, statement-id, object)
  - this means that we can compactly represent reification, which means we can also represent topic maps in a compact way
- **The model then works as follows**
  - it's a set of quads
  - the third element of each quad must be unique
  - you can't have the same quad twice with different statement IDs (no duplicates)
  - a quad identity cannot be used as a predicate

A little formality

- $\mathcal{I}$ **is the set of all identifiers**
  - an identifier is just an opaque token
  - it doesn't mean anything by itself, it just identifies something
- $\mathcal{L}$ **is the set of all literals**
  - these are data values like strings, integers, URIs, etc
- $\mathcal{A}$ **is the union of** $\mathcal{I}$ **and** $\mathcal{L}$
- **A model is a subset of (**$\mathcal{I}$ **x** $\mathcal{I}$ **x** $\mathcal{I}$ **x** $\mathcal{A}$**)**
- **q[n] produces the n'th element of q if q is a quad**
- **The paper introduces more helper functions**
  - these are used to define the return mapping from Q to TMDM and RDF

Representing Topic Maps in Q4

- **The following LTM**

  [lmg = “Lars Marius Garshol” = “LMG” / acronym]

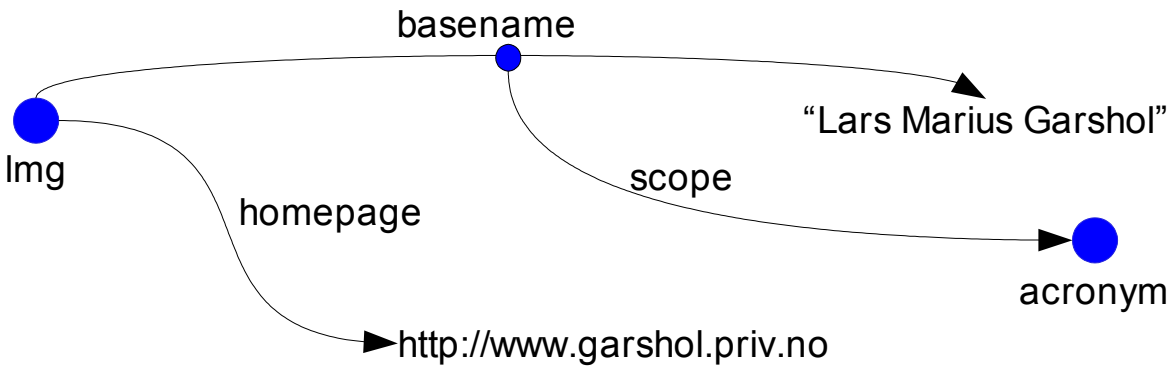
  {lmg, homepage, “http://www.garshol.priv.no”}

- **would turn into the following in Q4:**

  (lmg, basename, b1, “Lars Marius Garshol”)

  (b1, scope, _, acronym)

  (lmg, homepage, _, “http://www.garshol.priv.no”)



Is this it?

- **The bloat is gone**
  - just 22382 quads (4.6x TAO) for opera.xtm
  - with simple tricks, this is reducible to 11394 (2.4x TAO)
  - for mondial.xtm: 109737 quads, reducible to 43901 quads
- **Also, RDF and topic maps *mostly* have the same, natural, representation**
- **However, lots of difficulties remain**
  - we'll walk through and study the problems one by one

Problem #1: Associations

- **Associations in topic maps and RDF are not aligned here**
- **Binary relationships in RDF are a single quad**
  - (lmg, dc:creator, _, q)
- **Binary relationships in topic maps have one quad per role**
  - (assoc, type, _, created-by)
  - (assoc, creator, _, lmg)
  - (assoc, creation, _, q)

Solution #1

- **Treat binary relationships in RDF as having two “roles”**
  - (r, type, _, dc:creator)
  - (r, subject, _, lmg)
  - (r, object, _, q)
- **This is formally OK, but now we're back with a bloated model**
  - not good for implementation

Solution #2

- **Use “association templates”**
- **That is, for each (association type, role type 1, role type 2) combination, create an identifier, and use that**
- **Like this**
  - (Img, created-by-template, _, q)
  - (created-by-template, type, _, created-by)
  - (created-by-template, subject-role, _, creator)
  - (created-by-template, subject-role, _, creation)
- **This gets rid of the bloat**
  - however, association role reification is no longer representable
  - we can't define tolog on top of this, either

Problem #2: Language tags

- **String literals in RDF can have language tags attached to them**
  - strangely, these are RFC 3066 code strings rather than resources
  - effectively, this qualifies the literal, saying which language it's appropriate in
- **This is a common use for scope in topic maps**
  - therefore: represent language tags as though they were scope
  - turn each language code into a URI in a particular namespace
  - (because scope must consist of topics...)

Problem #3: Identifying URIs

- **In RDF, a URI can only be attached to a node in one way**
  - but it can mean two different things
- **In Topic Maps, a URI can be attached to a node in two ways**
  - the same two semantics still apply, of course
- **So, how to approach this?**
  - the naïve approach is to define two Q-properties
  - however, this causes a mismatch between RDF and Topic Maps in Q
- **Solution**
  - the property distinction really captures type information
  - capture the type separately, and use only one property
  - in RDF the type information must be added in order for a Topic Maps mapping to be possible
  - (this is in any case necessary to distinguish between names, occurrences, and associations)

Problem #4: Duplicates

- **Consider the following topic map:**
  - [fish = "Fish" = "Fisk" / norwegian = "Fisk" / swedish]
- **This gives the following in Q4:**

  (fish, TOPIC_NAME, _, "Fish")

  (fish, TOPIC_NAME, s1, "Fisk") ◄———

  (s1, SCOPE, _, norwegian)

  (fish, TOPIC_NAME, s2, "Fisk") ◄———

  (s2, SCOPE, _, swedish)
- **However, the two "Fisk" topic names give us duplicate statements (s1 and s2)**
  - this violates the "no duplicates" constraint
  - however, we can't merge s1 and s2, because in topic maps these are reifiable separately, and have separate variants
  - the same applies in RDF
- **This requires a radical solution...**

The Q model

The real thing, at last

The Q model

- **We go from quads to quints**
  - (subject, predicate, statement-id, context, object)
- **The "context" is used to represent scope in topic maps**
  - scope can consist of multiple topics
  - therefore, quints are used to attach the scoping topics to the context node
- **The same rules as before apply, but the no duplicate rule now takes context into account**
- **Our previous example then becomes**

  (lmg, basename, b1, c1, "Lars Marius Garshol")

  (c1, scope, _, Q, acronym)

  (lmg, homepage, _, U, "http://www.garshol.priv.no")
- **The size is now even more reduced**
  - 8551 quints (1.8x TAO) for opera.xtm
  - 43731 quints for mondial.xtm

Some example data

[lmg = “Lars Marius Garshol”]

{lmg, homepage, “http://www.garshol....”}

creator-of(lmg : creator, q : creation)

:lmg foaf:name “Lars M... Garshol” .

:lmg foaf:homepage “http://www...” .

:lmg dc:creator :q .

(lmg, basename, _, U, “Lars Marius ...”)

(lmg, homepage, _, U, “http://...”)

(lmg, t-creator-of, _, U, q)

(homepage, meta_type, _, U, occurrence)

(t-creator-of, meta_type, _, U, association)

(lmg, foaf:name, _, U, “Lars Marius ...”)

(lmg, foaf:homepage, _, U, “http://...”)

(lmg, dc:creator, _, U, q)

Applications of Q

- **Possible applications**
  - Dual RDF/TM implementations
  - Common model theory for both
  - Mechanism to apply RDFS/OWL inferencing to Topic Maps
  - TMQL/SPARQL mappings
  - etc
- **Actual applications**
  - mathematically formulated theory of scope (in progress; unpublished)
  - formal specification for tolog query language (in progress; accepted for TMRA'05)
  - efficient topic maps backends (in progress; very rough)

Specifying tolog on top of Q

- **We define a single predicate that is not visible in the language**
  - _q(subj, pred, id, ctxt, obj)
- **We then use this to define the built-in predicates**
  - topic-name($TOPIC, $NAME) :-
    _q($TOPIC, $P, $NAME, _, _),
    _q($P, meta-type, _, _, topic-name).
- **We map dynamic association predicates down to built-in predicates**
- **The same for dynamic occurrence predicates**
- **Finally, we need to define result sets, AND, OR, NOT, etc**
  - however, Q does the heavy lifting with the complex Topic Maps model

The scope theory

- **Defines two functions:**
  - b(M, i): models belief
    - produces the subset of the model that we believe if we believe *i*
  - d(M, i): models disbelief
    - produces the subset of the model that we believe if we only disbelieve *i*
- **These satisfy for all models M:**
  - b(M, $\mathcal{I}$) = M
  - b(M, Ø) = f(M, *, *, *, U, *)
  - d(M, Ø) = b(M, $\mathcal{I}$) = M
  - d(M, $\mathcal{I}$) = b(M, Ø)

The actual functions

- b(M, s) = { q in M | forall t in f(M, con(q), scope, *, *, *)[5] : t in s }
- d(M, s) = { q in M | not exists t in f(M, con(q), scope, *, *, *)[5] : t in s}

How to use RDFS/OWL with Topic Maps

- **The basic problem is that RDFS/OWL don't understand scope**
  - that is, if two statements Y and Z are needed to conclude X, but Y and Z are present with different scopes, then X is not necessarily valid
- **A possible solution is (possibly) to extend the scope theory**
  - the extension will be a function that creates a set of scopeless models, each of which corresponding to a scope in the model, and containing all the source quints that are known to be valid in that scope
  - obviously, some quints may appear in more than one submodel
- **Normal RDFS/OWL inferencing can then be done on each submodel**
  - this will extend the submodel with the quints known to be true in that scope for that submodel
- **The 48,000$ question is**
  - can this be done efficiently for all scopes at once, or just one?

More information

- Read the paper
- Email <larsga@ontopia.net>