

Implementing Inference Rules in the Topic Maps Model

Q.H.J.F. Siebers

July 21, 2006

Abstract

This paper supplies a theoretical approach on implementing inference rules in the Topic Maps model. Topic Maps is an ISO standard that allows for the modeling and representation of knowledge in an interchangeable form, that can be extended by inference rules. These rules specify conditions for inferrable facts.

Any implementation requires a syntax for storage in a file, a storage model and method for processing and a system to keep track of changes in the inferred facts. The most flexible and optimisable storage model is a controlled cache, giving options for processing. Keeping track of changes is done by listeners.

One of the most powerful applications of inference rules in Topic Maps is interoperability. By mapping ontologies to each other using inference rules as converter, it is possible to exchange extendable knowledge.

Any implementation must choose methods and options optimized for the system it runs on, with the facilities available. Further research is required to analyze optimization problems between options.

1 Introduction

For years, software development has been concentrating on supplying customers with data management systems. Slowly these systems are reaching their limits. Customers are changing their needs towards knowledge companies. New ways to capture business processes are required.

Topic Maps¹ provides a new solution for knowledge management and storage. New developments in the field of Topic Maps, like inference rules, are going to make knowledge derivation possible.

¹When the ISO model is referred, we use the term ‘Topic Maps’, when an instance of the model is referred, we use the term ‘topic map’.

1.1 Topic Maps

Topic Maps is an ISO standard that allows for the modeling and representation of knowledge in an interchangeable form, and provides a unifying framework for knowledge and information management [6, 4]. Topic Maps is based upon the characteristics of the index of a book, linking ‘topics’ to ‘sources’ by using page numbers (‘occurrences’). It extends the power and ideas of SGML, XML, semantic networks, RDF and Frames by adding more complex but intuitive solutions and possibilities [2, 14, 16, 12, 6].

TAO of Topic Maps

The three main concepts of a topic map are topics, associations and occurrences. With these three concepts as building blocks, it is possible to make a topic map. The term TAO comes from combining the first characters of these concepts [11].

Topics

The most important of these concepts is a topic, as suggested by the name of the ISO. As Steve Pepper said in his paper about the TAO:

A topic, in its most generic sense, can be any ”thing” whatsoever – a person, an entity, a concept, really anything – regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever.

Pepper, S. [11]

So, a topic is a concept in a specific domain, onto which someone wants to add knowledge or information. The topic itself is only a unique object that symbolizes a concept. The associations and occurrences store the information and knowledge available about that topic.

Associations

Associations are links between topics, binding them together in a specific meaning. A simple relationship between father and son can be expressed as an association with meaning ‘father-son-relationship’.

A topic map with topics and associations makes up a semantic network. It intuitively states logically based statements as ‘Peter is the father of Tom’, in which ‘Peter’ and ‘Tom’ are topics, related to each other by an

association ‘is the father of’. In this example it becomes very clear that saying they are related is only part of the relationship. Saying what role they play is equally important.

To be able to give directional meaning to an association, roles have been added to the topic map. With roles, an association is bi-directional, and can be viewed from the required position. In our example: ‘Tom’ plays the role of ‘son’ and ‘Peter’ the role of ‘father’.

Viewing associations as a web in a topic map can bring up deeper relationships. This is the basis for Inference Rules.

Occurrences

Not every property or bit of knowledge about a topic is required to be stored as an association. Therefore there are occurrences in a topic map. Occurrences contain a property of a topic. It can be as simple as a birthdate, or as complex as an URI [15] specifying some information source for the topic. Roughly said, an occurrence is a database field, containing some information about that record.

More properties

Of course there is more to a topic map than just the three concepts of the TAO.

A topic is only an object representing a subject. A subject can have several names, even in more than one language. Every name can have variations. Names and variations are another part of the Topic Maps model.

To be able to classify topics in groups and for defining hierarchies, the Topic Maps model has been extended with the possibility for typing topics. The type of a topic is a topic from the ontology of the topic map. Ontology types are also topics. They reside within the same topic map.

To uniquely identify objects in a topic map, public subject identifiers (PSI’s) were added to the model. PSI’s are URI’s linking to a description of the object, while the link itself is used for comparing and identification.

Another addition to the Topic Maps model is scoping. Characteristics of a topic can be scoped, giving these characteristics a context. Scoping is done by means of topics. This functionality is often used in multi-language systems.

And last but certainly not least, a technique called merging allows topic maps to be combined. A merged topic map is the combination of more topic maps where objects with same identifiers are merged to one single object.

Many more techniques and properties of topic maps are available, some of them restricted to a specific topic map system.

1.2 Tolog

To query a topic map, a language is required that can easily catch the intuitive approach of Topic Maps. On the other side this language has to be understandable for a topic-map system as well as for a human. The most commonly used language for querying topic maps is tolog [9], developed by Ontopia [8].

Tolog is a Prolog-like [3] language combined with SQL elements [13, 1], querying the data as a semantic web. Tolog is extended with predicates specifically designed for Topic Maps, making it able to query on both ontology and data.

1.3 Inference Rules

A basic goal of a topic map is to relate topics to each other in a way that uses associations that are as basic as can be. Therefore a grandfather relation between two persons is usually stored as two father-associations. This is as direct as one can relate these topics to each other without missing any knowledge about the associations. However, storing it as two father-associations brings the problem of this paper. Isn’t the knowledge about person ‘A’ being the grandfather of person ‘B’ something needed to be stored? Does this knowledge apply to other topics in the topic map? And most importantly, isn’t this knowledge also an association?

What are they?

Inference rules are a derivation of knowledge. They represent abstracted, generalized knowledge in a domain. When applied to the data they get converted into facts about the data. They provide a way to store generally known or abstracted knowledge in a single simple rule. This rule describes the conditions for a partition of the data being enriched. Inference rules are built upon existing networks or semantic webs, described and programmed in the same language of the data it enriches.

What do they do?

Inference rules define a way to create implicit associations between topics. These associations can only exist because of other associations or properties of the connected topics. Inference rules give a user the possibility to map knowledge onto an already defined set of data using its ontology as a basis, to build conditions with.

However they do not only extend the knowledge residing in the data, they also create a new piece of the ontology. This piece is a valid part of the ontology and can be used as any association type, meaning it can also be used in another inference rule.

Inferencing in tolog

When using a topic map in a tolog querying environment, a user may define inference rules in a separate file

which gets included on executing a query. The inference rules in these files are very basic. They resemble Prolog functions, consisting of a head with parameters and a body using the parameters to decide upon the parameters. With the use of variables, tolog, as Prolog, generates a list of possibilities for parameters supplied to the query. This process is the same for inference rules.

But why keep the inference rules in a separate file? They create additional knowledge and ontology items for a range of specific topic maps. They are part of the data and of the ontology of these topic maps and therefore they should be stored in the topic map itself, making it possible to exchange topic maps between systems, without losing any additional knowledge.

Problems with Inferencing

Because inference rules define additional knowledge in a topic map, some predicate query language like tolog is needed. And because an inference rule's body is usually in the same language, the body consist of predicates. So every loophole in the predicate language will also exist in the inference rules. Most commonly these problems are recursive inference rules and undecidable inference rules. Trying to solve NP-problems will lead to undecidability. Any implementation should have some way of dealing with these problems without crashing the system.

1.4 Research question

To be able to make an implementation of inference rules in Topic Maps, it needs to be clear what the goals of this implementation are. So the question becomes: "What do we want to do with Inference Rules?". There is no single answer to this question; every user has its own wishes and goals of implementations.

There are however some goals that can be set, covering the most powerful and extendible features of inference rules. The most simple of these goals would be the possibility to query for all associations, including inference rules.

Another powerful goal is the reusability of inference rules. This means that an inference rule must be able to use other inference rules as conditions, making it possible to build a hierarchy of inference rules.

Exchanging inference rules between topic maps can be considered another powerful goal of implementing inference rules in Topic Maps.

Taking these goals as a basis, considering any implementation extendible enough for any other goal, a picture of a general implementation becomes more clear.

The remainder of this paper is structured as follows. Section 2 describes the main requirements needed for any implementation. Section 3 proposes a theoretical implementation model. Section 4 discusses a major advantage of inference rules in Topic Maps. Section 5 presents the conclusions and supplies a view to the future.

2 Requirements

Using the goals of any implementation as a starting point, we can define what is needed for an implementation, and specify a guide for meeting these requirements.

2.1 Syntax

To be able to re-use and distribute the inference rules along with a topic map, we need a way of storing them inside the topic-map files. This can be done fairly easy by extending the commonly used syntax standards as LTM [7] and XTM [10] to accept inference rules. Building on the Topic Maps model, the best approach is to create a topic as the type of the associations, and extend this with zero, one or more inference rules. An example of how this could be done for LTM:

```
[grandparent-relation = "Grandparent relation"
  = "has grandchildren" / grandparent
  = "has grandparents" / grandchild
]
[grandparent = "Grandparent"]
[grandchild = "Grandchild"]
```

```
grandparent-relation($GP : grandparent,
  $GC : grandchild) :-
  parent($GP : parent, $MID : child),
  parent($MID : parent, $GC : child).
```

First an association type is defined, described by a topic with a general name. Supplying names scoped by the roles results in the possibility for outputting directional sentences. Next the role types are defined, described by topics. Finally, inference rules can be added to the association type.

The association type and the predicate in the head of the inference rule are equal, and should generate one single association type topic on parsing. So any association between two topics can be made explicitly by creating an association in the topic map, or implicitly by an inference rule. This is exactly what is wanted of course, because when using identifiers and exchanging topic maps we want the associations and the inference rules to be the same subject.

In the example above we see the use of roles in the inference rules. This does not only clarify the meaning of the rule, it also makes it easier to process the rule when matching. This feature will be explained more deeply in section 2.3.

2.2 Storage

After loading a topic map from a file, an internal storage in the memory of a system that supports inference rules is required for storing both inference rules and inferred facts.

When thinking about this, one can see that there are two extreme and opposite ways of dealing with this problem, based upon how much is kept in storage.

Store all

The first possible solution is to keep every inferred fact stored in memory. This could cause a tremendous amount of associations, flooding the system. When considering big topic maps this could lead to an in-memory storage problem. Also, a way of keeping track of changes would be required to maintain the inferred facts. On the other hand, this solution would not need much calculation time after initialization.

On the fly

The opposite of storing everything is to store nothing at all. This would mean that every time an inference rule is used in a query, all the facts have to be derived again. This solution would also limit the usage of the query results. There would be no need for keeping inferred facts up to date, and there would not be a storage overhead.

Caching

Combining the positive characteristics of both extremes gives us the cache solution, which will only keep some part of the inferred facts in storage. This solution has a low storage overhead but is time effective. It requires a mechanism of determination which inference rules need to be inferred into a cache, and which not. It also requires a mechanism to keep track of changes in the inferred data in the cache.

2.3 Listeners

When using the cache and store-all solutions a way of keeping the inferred facts valid is required. What is wanted of course is to only update the inference rules that are changed. The answer to how this can be done is located in the body of the inference rules. The body consists of predicates. These predicates represent the types of associations, occurrences, names, topics and other inference rules that decide when the inference rule derives a fact. When a change is made to the topic map, the type of the object(s) affected by the change play the most important role.

A fact, derived from an inference rule can only change when facts in its body's predicates change. Therefore, the fact can only change when a change effects a type that is used as a predicate in the inference rule's body. To create the most direct action on a change, it is the type of the object changed that must activate the re-evaluation of certain inferred facts.

To maximize the efficiency of re-evaluating inference rules, a minimum amount of inferred facts has to be re-evaluated. Finding only the inferred facts that a change might have effect on, can be done by applying the following steps.

Step 1: Find inference rules with bodies containing the predicate that is to be changed

As stated earlier, a change can be mapped to a predicate in the body of an inference rule. Only the inference rules with that predicate are needed. Note that the predicate can also point to another inference rule. So a change can effect inference rules via other inference rules.

Step 2: For each body, for every occurrence of the predicate, match the values to the other parameters of the body

When a predicate is used more than once in a body, then matching needs to be done for every occurrence separately. Matching the parameter values of the predicate on the other predicates in the body leads to a query with some free variables.

Step 3: Execute the body as a query, revealing all possible changed inferred facts. Keep these available

Querying on the free variables that remained in the body after matching to the change, leads to all possibilities for all the free variables. Extracting the parameters of the head out of the query result supplies a direct link to inferred facts that need re-evaluation.

Step 4: Apply the change

The change can only be applied at this point because the original situation was needed to find the facts in need of re-evaluation.

Step 5: Redo step 2 and 3 with the new values to infer new facts

Any change could lead to new facts. These facts can be inferred by matching the new values to the body of the inference rule and querying the body. Results from this query can be added as inferred facts, if they do not already exist.

Step 6: Apply the headers obtained in step 3 to the inference rules to evaluate the fact by querying the body

Existence of a fact is determined by the inference rule's body matched to the parameters from its head. When a queried matched body results in any number of results, the fact is still valid. Failing to return results is the falsification of the fact.

Example

Consider the topic-map contents in Figure 1.

IA1 is an inferred association inferred by inference rule IR1. T1, T2 and T3 are topics, A1 and A2 are associations of types AT1 and AT2, respectively. The R's are the roles the topics play in each association. IR1 is defined as:

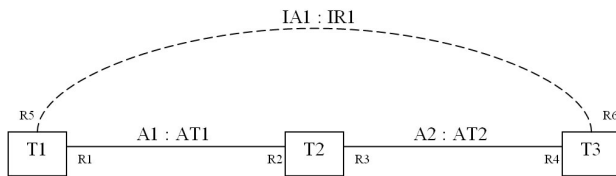


Figure 1: Example topic map contents.

```
IR1 ( $X : R5, $Y : R6 ) :-
  AT1 ( $X : R1, $Z : R2 ),
  AT2 ( $Z : R3, $Y : R4 ).
```

Consider deleting association A1. The type of A1 is AT1. AT1 is used as predicate in IR1, so IR1 need to be examined more detailed. AT1 only occurs once in IR1. Applying step 2: \$X is matched to T1 and \$Z is matched to T2, which leads to:

```
AT1 ( T1 : R1, T2 : R2 ),
AT2 ( T2 : R3, $Y : R4 ).
```

Running this body as a query will result in one match: \$Y to T3. This means that there is one inferred fact, connection T1 with T3. This fact need to be re-evaluated after applying the change.

After applying the change, step 5 can be executed. The new values are the old values in this example, so the body would be matched to:

```
AT1 ( T1 : R1, T2 : R2 ),
AT2 ( T2 : R3, $Y : R4 ).
```

However, the first statement is no longer true. This means this query will not return any results. Therefore no new facts can be derived from the change in this example.

Finally step 6 can be applied. There was only one fact in need of re-evaluation. Using the results from the query in step 3, the inferred fact can be re-evaluated by matching the parameters in the head:

```
IR1 ( T1 : R5, T3 : R6 ) :-
  AT1 ( T1 : R1, $Z : R2 ),
  AT2 ( $Z : R3, T3 : R4 ).
```

Note that only the parameters from the head get matched. After querying the body, an empty result is returned. This means that the inferred fact is no longer valid and needs to be deleted.

Changing one of the role players of association A1 would lead to the same processing, with exception to step 5. The new role player could create a new fact.

2.4 Cache control

To limit the amount of overhead in storage, a system must define a way deciding on cache questions. If an inference rule is cached, then it must cache all its inferred

facts to maintain completeness. This implies that general inference rules that create a lot of facts require a big cache. There are several easy ways of dealing with this.

Maximized cache

When allowing only a fixed maximum of inference rules and inferred facts of these inference rules, this would limit the cache. This will work in situations when inference rules are used more than once sequentially, but will not work when more inference rules than the maximum are used sequentially.

User defined

Letting the user define which inference rules to cache, and which to process on-the-fly can limit the cache. However, the user will need to know if a specific inference rule can or must be cached or processed on-the-fly. Changes in the topic map can also change the need for on-the-fly processing.

Calculating vs storing

The efficiency of any caching solution is dependent on the type of system and topic map in question. In general, testing multiple solutions on a system is the best way to reveal what solution is to be used. The user has to decide between more calculating or more storage, dependent on the user's cost analysis.

Clearing cache space

Removing an inference rule with inferred facts from cache comes back in most solutions. It is very important to keep track of how many and which inference rules are cached. As long as they are in the cache, they will get updated because of the listeners. When, for any reason, an inference rule and its inferred fact has to be removed from the cache it is very important that all inferred facts and their listeners are removed. Note that the system must register the inference rule removed from cache as being un-cached. This will keep the cache complete and clean. The listeners can not work properly without this completeness.

3 Implementation

To accommodate the cache solution for storing and processing inference rules, a model is needed that keeps track of changes in the topic map. This model would need to keep track of changes in loose topic map objects, as well as system-wide events as additions or deletions. Furthermore, this model needs to support inference rules referring to other inference rules in their body.

3.1 On adding objects

When an object is added to a topic map its type plays the major role. If this type is mentioned by a predicate in the body of an inference rule, this inference rule has to

be re-evaluated to check if new facts can be derived from it. Note that this only applies to inference rules that are already cached. Inference rules that are not cached have not been queried yet, so the addition will be processed on querying.

When the added object triggers the inference rule to create a new derived fact, then a listener should be added to the new object giving the inference rule and the derived facts a straight and fast way to respond to any changes or deletions of the object.

On adding inferences

Adding new inference rules to the topic map is different from adding objects to the topic map. The inference rule itself can be added after checking for known problems in the body, so it will be processed and cached on the first query that uses it. But the system needs to create listeners to and from the new inference rule connecting it with the objects represented by the predicates in its body. On this way, when something changes in one of the objects represented by the predicates, the new inference rule gets notified of the change and acts accordingly.

3.2 On deleting objects

On deletion of an object from the topic map, the system must notify all connected objects of this change. By supplying the type of the object being deleted, the notified object can easily decide on the action it needs to take. When this notification is sent to an inferred fact, it can partially decide on its own if it should still exist. In this way, no re-evaluating is needed for facts that lose one of their role players.

On deleting inferences

Logically, when an inference rule is deleted, all the inferred facts in the cache that were created by this inference rule need to be deleted. Deletion can only be possible if there are no other inference rules using this inference rule. More implementations for dealing with this are possible. Not accepting the deletion or deleting all linked inference rules are some of the possible implementations.

3.3 On changing objects

A change in a topic map is usually only possible for names, occurrences, scopes, and identifiers. In most implementations of a topic map system, changing an association leads to its deletion and recreation. A topic is nothing more than an internal identifier for some subject and therefore should never have the need to be changed.

For the rest of the topic map objects, a change can be processed in the same way as a deletion. All listeners must be notified of the change, and in most cases this will lead to re-evaluation of inferred facts.

3.4 On querying

The essential part of the cache solution is located at querying of the topic map. When inference rules are used in the query, the system needs to check if they have been stored in the cache, or not, and if they should be. When they are in the cache, the system can assume the facts in the cache as accurate and up-to-date and use the inference rule as an association type for the inferred facts.

If an inference rule is not yet in the cache, the system should infer the inference rule. On this inferring, new facts are derived from the inference rule, which must get a listener that will notify the inference rule on any changes, unless the inference rule must be processed on-the-fly.

Also, the system should note that the inference rule has been inferred to the cache for later querying, so that it will not get inferred time after time while being in the cache.

3.5 System-specific problems

To accommodate users, most topic map systems contain query predicates on a meta level or serving a shortcut purpose. These predicates can also be used in the body of an inference rule. The problem with these predicates is that they are not topic map objects, and therefore listening for changes is a problem. Every implementation for a specific system much make choices on which and how to support these predicates.

4 Interoperability

Expanding knowledge in a topic map is only a part of the usability of inference rules stored in a topic map. Another strong advantage of inference rules is strong enhancement of interoperability between topic maps.

Interoperability is the ability of products, systems, or business processes to work together to accomplish a common task. The term can be defined in a technical way or in a broad way, taking into account social, political and organizational factors.

In the context of Topic Maps, interoperability is the problem of combining and sharing more than one ontology to exchange knowledge. When implementing inference rules into a topic map, it becomes possible to solve this problem for Topic Maps. Creating a layer of inference rules to map one ontology to another is called ontology mapping.

4.1 Identifying

The start of the solution towards interoperability lies in the main structure of Topic Maps. Topics uniquely represent subjects via identification. On merging of two topic maps, the key to success lies in making equal

subjects be equally identified. Setting PSI's from both topic maps to both topic map's topics creates the perfect merging. The merged topic map contains single topics representing unique subjects, identified by PSI's from both original topic maps.

So, interoperability in basic topic maps is nothing more than getting PSI's aligned and merging them. However, this does not supply solutions for combining inference rules. Neither does it supply a solution for exchanging knowledge without merging.

4.2 Identifying inference rules

When inference rules are stored within the topic map, inference rules can be given unique identifiers as if they were associations. This makes it possible to merge them with other topic maps. Inference rules from one association can simply be added to the other topic map's association, given the system checks the meaning of the body for double definitions.

4.3 Merging-less exchange

A merging-less exchange can be viewed as a query to an external topic map. This external topic map's ontology can not always be using the same ways of storing knowledge. This is where the inference rules come in. The direction and location of processing of the exchange can be split up in two separate processes.

Externalization

The request is made using an inference rule specifying how to create internal ontology specifications out of external ontology specifications. A topic map could request a non-existing association, on which the external system could derive the meaning of the body in its ontology and reply.

Internalization

The request returns the way the external ontology describes a requested fact. The returned inference rule, translated into internal ontology, extends the ontology. Derivations of this inference rule, can lead to new types and even to context rules.

A request for a definition of brother-of would return an inference rule specifying there has to be a mother and a father for a topic to have a brother. This context rule is derived knowledge, extending the topic map.

TMRAP

To enable interoperability for manual usage in a controlled basic topic map environment, Ontopia has been developing Topic Maps Remote Access Protocol (TMRAP) [5]. TMRAP provides the user the possibility to import topic map objects out of external topic maps. Extending this with inference rules will enable the possibility for importing derived or derivable knowledge from external topic maps.

5 Conclusions and future research

Inference rules stored in topic maps are a very powerful tool for many applications, yet to be discovered. This paper supplied a model as a guide, to further these discoveries.

The implementation of inference rules in the Topic Maps model is very dependent on the system and topic maps it is required to support. When implementing this into a system, many considerations have to be made in advance of the actual implementation.

The model proposed in this paper yields opportunities to optimize implementation based on its needs. Choices on storage are mainly decided by available facilities. This also holds for cache implementations.

As for optimization questions, more research is needed on most of the model's options. Especially the listener structure must be as compact and efficient as possible to limit and minimize computational costs.

Interoperability is one of the features made easier by inference rules. More research is needed to uncover more fields of application possibilities, as well as for defining a more structured, generalized interoperability procedure for topic maps.

Storing inference rules in topic maps and expanding their power are main requirements for further study and development on both ontology mapping and interoperability.

Towards OWL/RDF

Currently RDF combined with OWL gives a system the possibility to extend the knowledge stored in RDF [17]. Topic Maps does not have support for OWL. However with inference rules and storage of these in the topic map, possibilities for OWL-like development are surfacing.

Artificial Intelligence

The possibilities of inference rules in topic maps also extend the possibilities for using Topic Maps as a basis for new development in Artificial Intelligence. The power of automatically deriving new facts out of a set of knowledge can push research into Artificial Intelligence to a new level.

New standards

The Topic Maps community is working hard on cooperation with other organizations, combining standards for more powerful solutions for the future of topic maps. Committees are working hard to keep the ISO standards used by Topic Maps up-to-date to new developments. New storage syntaxes are under creation, as well as protocols for basic interoperability on user levels.

Acknowledgement

Part of this research has been performed in cooperation with Morpheus Software, Ontopia and the Universiteit Maastricht.

We thank P. Kruijzen of Morpheus Software for the inspiration leading to this research and his guidance in the Topic Maps world.

We would like to thank G. Hopmans of Morpheus Software for his remarks and guidance on the ISO side of Topic Maps.

We would also like to thank L. Garshol of Ontopia for sharing his Topic Maps expertise and his valuable time to the benefit of this research.

Finally, thanks goes to J. Uiterwijk of the Universiteit Maastricht for his guidance and remarks with regards to the creation of this paper.

References

- [1] Chamberlin, D.D. and Boyce, R.F. (1974). *Sequel: A structured english query language*. *International Conference on Management of Data*, pp. 249 – 264.
- [2] International Organization for Standardization (1986). *Information processing - text and office systems - standard generalized markup language (sgml)*. Technical Report ISO 8879, International Organization for Standardization, Geneva, Switzerland.
- [3] International Organization for Standardization (1995). *Information technology - programming languages - prolog*. Technical Report ISO/IEC 13211, International Organization for Standardization, Geneva, Switzerland.
- [4] International Organization for Standardization (2000). *Information technology - sgml applications - topic maps*. Technical Report ISO 13250, International Organization for Standardization, Geneva, Switzerland.
- [5] Moore, G. (2004). *Topic maps remote access protocol 0.2*. <http://www.jtc1sc34.org/repository/0507.htm>.
- [6] Ontopia (2001). *What are topic maps?* <http://www.ontopia.net/topicmaps/what.html>.
- [7] Ontopia (2006a). *Ltm: The linear topic map notation*. <http://www.ontopia.net/topicmaps/ltm.html>.
- [8] Ontopia (2006b). *Ontopia: The topic map company*. <http://www.ontopia.net>.
- [9] Ontopia (2006c). *Tolog*. <http://www.ontopia.net/topicmaps/materials/tolog.html>.
- [10] Pepper, S. and Moore, G. (2001). *Xml topic maps (xtm) 1.0*. <http://www.topicmaps.org/xtm/1.0/>.
- [11] Pepper, S. (2000). *The tao of topicmaps*. <http://www.ontopia.net/topicmaps/materials/tao.html>.
- [12] Sowa, J.F. (2000). *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks / Cole, Pacific Grove.
- [13] Wikipedia (2006). *Structured query language*. <http://en.wikipedia.org/wiki/SQL>.
- [14] World Wide Web Consortium (2006a). *Extensible markup language (xml)*. <http://www.w3.org/XML/>.
- [15] World Wide Web Consortium (2006b). *Naming and addressing: Uris, urls, ...* <http://www.w3.org/Addressing/>.
- [16] World Wide Web Consortium (2006c). *Resource description framework (rdf)*. <http://www.w3.org/RDF/>.
- [17] World Wide Web Consortium (2006d). *Web ontology language (owl)*. <http://www.w3.org/2004/OWL/>.