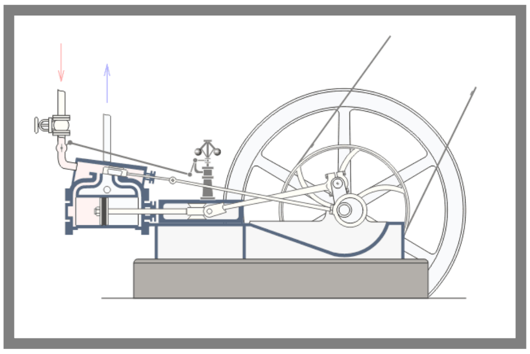
Making Linked Data Reasonable using Description Logics, Part 2

by Mike Bergman - Sunday, February 15, 2009

http://www.mkbergman.com/476/making-linked-data-reasonable-using-description-logics-part-2/



Good Design is Based on the *Work* to be Done

In Part 1 of this series, I advocated the placement of linked data in an ABox construct from description logics [1] based on a separation of concerns argument. Actually, this broader position arose from close inspection of an earlier table on TBox and ABox purposes and roles that I had assembled from the literature.

That table synthesized TBox and ABox purposes and roles from the citations listed in Back to the Future with Description Logics, though was based for the most part on the *The Description Logic Handbook* [2]. As we continued to look further at the assignments in that table we saw a few issues:

- First, some of the assignments seemed to be in error (as likely due to my own misunderstandings when making the assignments than anything else)
- Second, the use of my phrase "purposes and roles" in describing the assignments seemed too mushy: what ***work*** actually could occur within a 'box' seemed to be a more precise way of thinking and evaluating the assignments
- Third, two columns were inadequate: some ***work*** requires both the ontology and instances in order to be performed (that is, the work occurs at the conjoined layer of the knowledge base), and
- Last, some useful or necessary ***work*** may not even occur in whole or part within the confines of either the ABox or TBox.

Work, of course, is what our computers do for us on the data. Properly identifying and isolating these work tasks is a good starting point for teasing out architectural and schema design.

These perspectives allowed Fred Giasson and me to re-evaluate this earlier assignments table (see earlier version), as shown below. Note that some items have been moved to a different column (shown in blue, all of which were formerly in the 'ABox') and some have been added and may be external (shown in green):

Work Tasks for an 'Open World' Knowledge Base

TBox	TBox <--> ABox	ABox
• *Definitions* of the *concepts* and *properties* (relationships) of the controlled vocabulary • *Declarations* of *concept axioms* or *roles* • *Inferencing* of relationships, be they transitive, symmetric, functional or inverse to another property • *Equivalence testing* as to whether two classes or properties are equivalent to one another • *Subsumption*, which is checking whether one concept is more general than another • *Satisfiability*, which is the problem of checking whether a concept has been defined (is not an empty concept) • *Classification*, which places a new concept in the proper place in a taxonomic hierarchy of concepts • *Logical implication*, which is whether a generic relationship is a logical consequence of the declarations in the TBox • *Infer property assertions* implicit through the transitive property	• *Entailments*, which are whether other propositions are implied by the stated condition • *Instance checking*, which verifies whether a given individual is an instance of (belongs to) a specified concept • *Knowledge base consistency*, which is to verify whether all concepts admit at least one individual • *Realization*, which is to find the most specific concept for an individual object • *Retrieval*, which is to find the individuals that are instances of a given concept • [*Identity relations*, which is to determine the equivalence or relatedness of instances in different datasets] • [*Disambiguation*, which is resolving references to the proper instance]	• *Membership assertions*, either as *concepts* or as *roles* • *Attributes assertions* • *Linkages assertions* that capture the above but also assert the external sources for these assignments • *Consistency checking* of instances • *Satisfiability checks*, which are that the conditions of instance membership are met

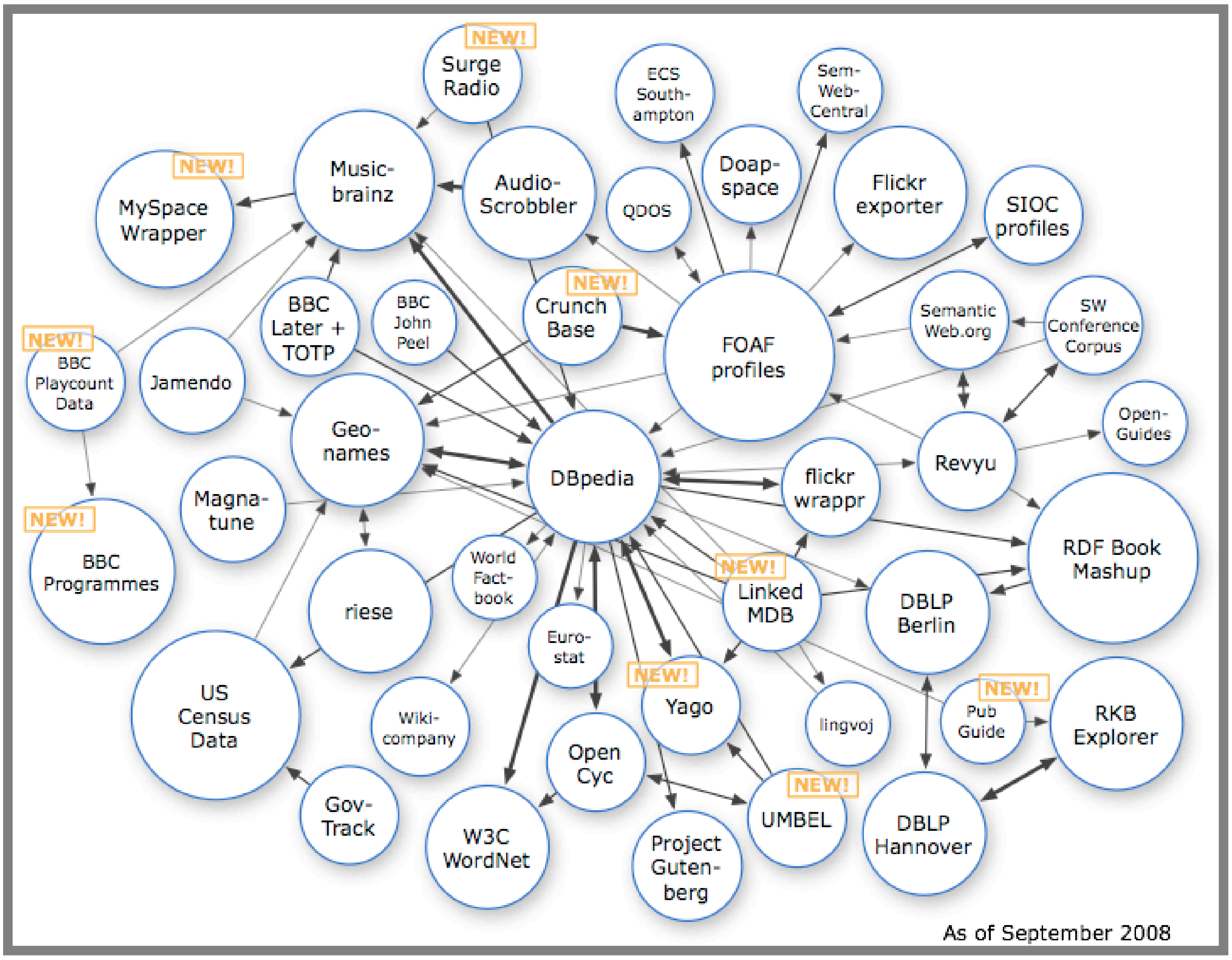
Blue = moved
Green = added

As the table now shows, the TBox is where the reasoning work occurs, the ABox is where assertions and data integrity occurs, and knowledge base work in the middle (among other aspects) requires both.

The State of Linked Data

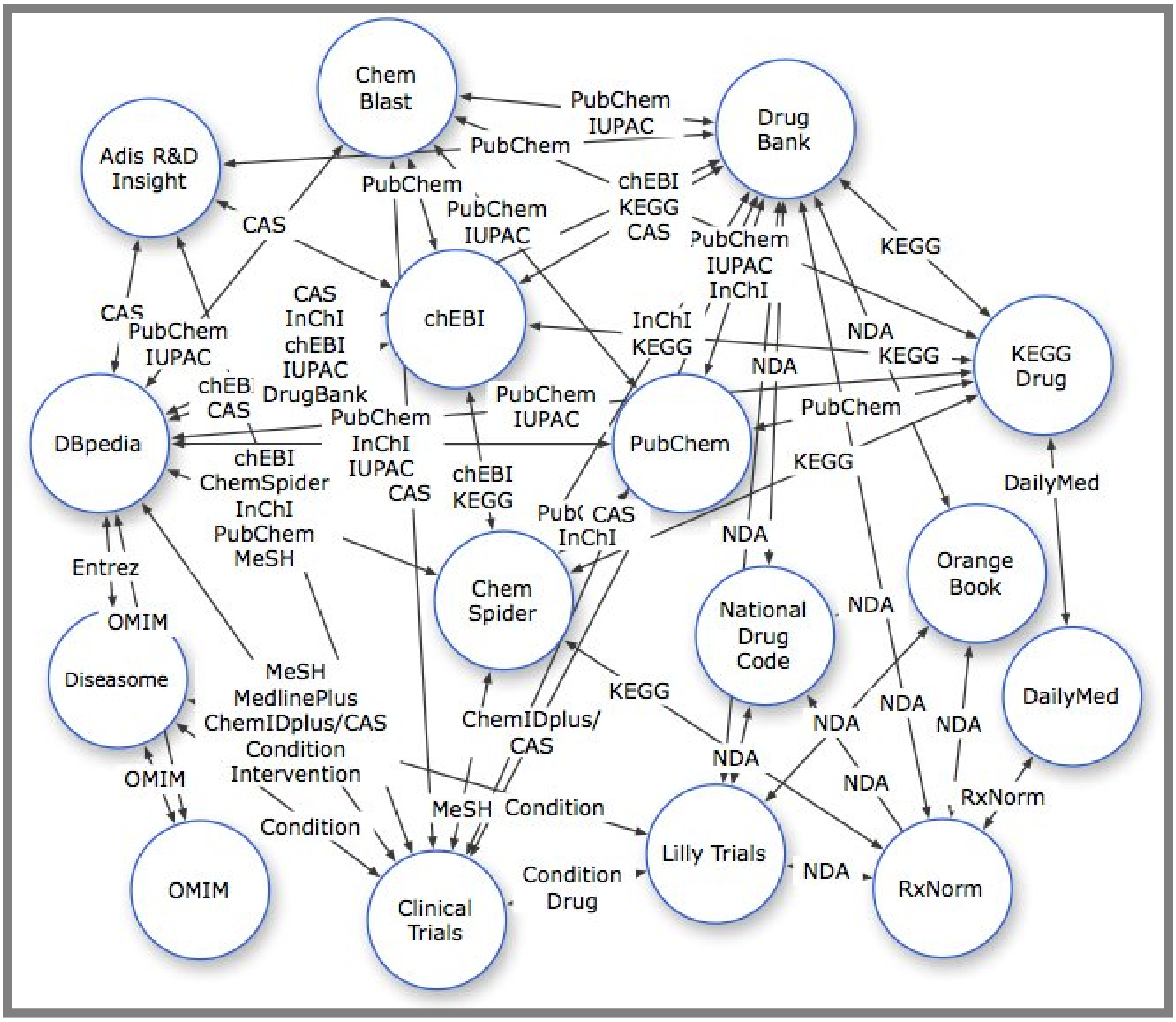
As Part 1 discussed, linked data in its current form, which are mostly instance records, most closely resembles ABoxes. The linkage in current linked data occurs via asserted relations and identities with things in other datasets (or sources). This might be the references to the objects themselves or to the predicates (properties) that describe the nature of the asserted relationship. The identifiers for these references are URIs, which are often external to the originating dataset.

This construction has led to the now well-known 'LOD cloud' for publicly accessible open linked data:



Each of the arrows on the diagram reflects an external linkage based on these property or object identity assertions.

This same approach can be applied as well for specific domains. This example from the LODD ('linking open drug data') "cloud" shows [3] a similar "linked" structure; also note the central node of DBpedia, also shared with the previous diagram:


Chem Blast
Drug Bank
Adis R&D Insight
chEBI
KEGG Drug
DBpedia
PubChem
Chem Spider
Orange Book
National Drug Code
DailyMed
Diseasome
OMIM
Clinical Trials
Lilly Trials
RxNorm
PubChem
IUPAC
CAS
chEBI
KEGG
InChI
DrugBank
ChemSpider
MeSH
Entrez
NDA
DailyMed
MedlinePlus
ChemIDplus/CAS
Condition
Intervention
Drug
RxNorm

Part of my argument in Part 1 was for the linked data community and its advocates to view its role via an ABox perspective. I further argued that this could help simplify schema and vocabulary design for publishing linked data, helping to promote faster and broader uptake.

We can now take these arguments a bit further by adding the perspective of useful ***work*** that can be performed within this ABox framework, building upon the table above.

Looking at linked data from the perspective of ***work*** makes sense as we see the scale of linked data grow. Besides the sheer increase in numbers arising from that growth, newer publishers may have fewer skills and less conformance to best practice. Questions as to quality and erroneous mappings now appear more frequently on the support forums and mailing lists.

Linked data -- or even more generically, distributed or federated data systems -- have particular needs for *search*, for checking *identity relations*, and for *disambiguation* that arise from their distributed nature. These are challenges, and were not generally topics in the initial discourse around description logics. On the other hand, I think it fair to say that linked data has already shown its benefits in *browsing* and *discovery*, and for linked data destination sites with SPARQL endpoints, benefits in *local search*.

Where Linked Data *Works*

The greatest strength of linked data, imo, is that it is showing the way to how the Web can become a global, interoperating database, the Web of Data. Standards are being applied, practices are being worked out, and actual data is being exposed. Visibility and the realization is growing. This is the real story -- and an important one -- and frankly a massive development that is a mere two years old.

Fundamental to this growth has been the superb flexibility and simplicity of RDF as a data model. No matter what else, the benefits of RDF and linked data have a signal that now exceeds the noise. I believe it can be comfortably asserted that the next generation of information systems will be built on this and REST, the combination of which has been called Web-oriented architecture (WOA).

OK Browsing and Discovery

If you know the locations and ***if*** you know the format and (sometimes) ***if*** you know the syntax, you can see and discover much from linked data on the Web. If you can get beyond these start-up hurdles, you can now start swimming in a sea of linked references.

Granted, sometimes you don't know what those cryptic link references mean; other times those links may take you to dead ends or silly stuff. I think it is fair to say that the community that is exposing this linked data stuff would acknowledge that their user interfaces and work flows are not yet intuitive or often easily grasped.

But, bear with this for a moment. Try a couple of these links:

- http://dataviewer.zitgist.com/?uri=http://mkbergman.com/me/

- http://dbpedia.org/resource/2007_Big_12_Women's_Basketball_Tournament

(BTW, any URI will work so long as it has RDF in the correct, servicable form; a challenge unto itself! [4])

Just like we first discovered the magic of moving from hyperlink to hyperlink in the initial Web, we are now beginning to see the process of moving from hyperdata to hyperdata with regard to structured and organized information.

Browsing and "stumbling upon" new cool stuff is the current main strength of linked data; again, assuming you are given some tips about how to start. Here are some linked data browers that might help you on this discovery journey, often with useful accompanying help and link examples: OpenLink Data Explorer or Zitgist Data Viewer or Marbles or DISCO or Tabulator. Don't be surprised if some of them break or don't work! (Again,if you don't know where to begin, try issuing one of the two link queries above after the 'url' or 'uri' part.)

Good Individual Dataset Querying

If you are a bit more experienced, some of the linked data source sites can be queried directly. The language used is SPARQL, which bears many resemblances to SQL for standard relational databases.

Generally (unless you are an expert), you will issue your queries over the local database at the target site. To see and test some examples, first bring up one of the many SPARQL clients, for example this one presents a rather complete overview and demo environment:

- http://demo.openlinksw.com/sparql_demo/ -- try it with both local and remote options.

A couple of other options with pre-loaded queries include:

- General: http://librdf.org/query (scroll down and use the Run this query links)
- Bio2RDF: http://bio2rdf.wiki.sourceforge.net/Demo+queries (and use the Try it out! links)

The SPARQL facility is really quite powerful and elegant; for the knowledgable user or system, this query syntax can also be applied across linked datasets. (Indeed, SPARQL is a key component in the background to Structured Dynamics' services.) SPARQL itself is a deserving topic in its own right.

OK Property and Object Search

There are a few RDF search engines that can do property, object or ontology searches, including Falcons (IWS China), Sindice (DERI Ireland), Watson (KMi), Semantic Web Search Engine (SWSE) (DERI Ireland) and Swoogle (the Ebiquity Group at UMBC). My personal favorite is Falcons.

We also are beginning to see structured data and attribute information creep into major search engines, often with little fanfare or notice; however, these are structured supplements to standard search results and not directly searchable alone. And, finally, there is a class of semantic search engines that do provide structured search, but not directly relevant to linked data (that is, they add structure and semantics to conventional search results). Prominent examples include Powerset (Microsoft), Hakia, True Knowledge and Cognition.

Where Linked Data is Not *Working*

In general, user interfaces have not been strong points for linked data. The emphasis, after all, has been more on the principles of linking and getting more data exposed.

More critical, however, from the standpoint of making the compelling case for an evolving Web of Data, is the weak tools support as the number and breadth of sources goes up. These limitations include *full-text search*, which requires performance equivalent to standard search engines; a lack of *identity testing*; and -- increasingly -- the need for *disambiguation*.

Only Partial Full-text Searching

Though there are linked data datastores (generally called "triple stores") for RDF that do full text search, they do not perform as well as dedicated text search engines. Depending on scale or hardware choices and configuration, these issues may often be overcome and not directly apparent to end users.

A more interesting concern and weakness arises from the nature of linked data itself. For example, here is the triple RDF statement for Mike Bergman knows Fred Giasson:

```
<rdf:resource="http://mkbergman.com/me/" foaf:knows
      rdf:resource="http://fgiasson.com/me/">
```

Now, what happens if I enter the search phrase "Fred Giasson" into my search box? Will this record appear?

The answer, often times, is no. And the reason it may not is that the "thing" of "Fred Giasson" is now represented by the resource URI of "http://fgiasson.com/me/" that does not contain the "Fred Giasson" search string.

In fact, one of the first observations that new users exposed to searching link data make is, Where are all of the results? Because, you see, they expect to see the same complete listing of results that would be returned by a conventional search engine.

The reason this occurs is that one of the very strengths of linked data -- to give unique resources a unique Web ID or URI -- is also what abstracts the resource from its literal string descriptor.

Now, this issue can be overcome, though to my knowledge no one is doing it. First, for a given resource, you can get all the triples where the object is a text "string" and index them. Then, for a subsequent resource, you can get all of the triples where the object is a resource and, if it is of the right type indicating a label, we then try to find the matching property for that resource.

This is generally not work done by triple stores. We are in essence trying to trace back from a URI identifier to its literal text descriptor (if it has one) and substituting the text descriptor into the search index such that it can be found during a full-text search (whew!). That sounds hard and like a lot of ***work***. Indeed it is.

However, as time goes on, this is likely functionality that users will demand. And, because the nature of the work is really text search and not triple stores or inferencing based from a graph, it may require special processing at ingest and the use of dedicated text engines.

No Identity Testing

Remember what we said earlier about linked data and its relation to instances and the ABox? It can ***assert*** it is related to or has properties or attributes of one kind or another, but from the information contained in the instance record itself we can not determine if those assertions are true. We can believe or trust some sources more than others as being more *authoritative*, but even for authoritative sources we may want to test the assertion.

Perhaps this is less of a problem when the linked datasets come from a relatively small community, as has been the case. But that is rapidly changing, and how do we begin to test *identity relations*?

Again, to my knowledge, identity testing is not yet being applied to linked instance data; we are relying on assertions and trust, each confounded by differences of opinion as to what asserting "identity" even means. Identity testing is a good example of a possible service or component that could either reside external to the TBox, or could use the concept graph at the TBox level to aid its logic.

Identity relations would check the assertions of relatedness made at the ABox level. At its simplest level, this may only be a lookup service for synonyms or aliases (or what we more broadly call 'semsets' in UMBEL). In more complex or comprehensive forms it could, for example, do a check for similar instances across the instance base (all ABoxes) using techniques similar to disambiguation. In this manner, for example, `owl:sameAs` could now be better determined, and under well defined conditions that users could understand and then accept or reject. Disjointedness and similarity are two additional identity checks possible.

Identity checking could thus better work across disparate datasets and could have a relatively simple and optimized index structure. This would be useful for real-time access; newly discovered identities could be separately slip-streamed and later evaluated more rigorously.

No Disambiguation

Disambiguation is another component that could similarly reside internal or external to the standard system. Like identity relations, it might rely on concept graph information but its logic need not be explicitly modeled at the TBox level.

Disambiguation basically tries to test whether Joe Farmer the farmer is the same or different than Joe Farmer the truck driver. TBox information can certainly aid this -- such as whether agricultural concepts or attributes are in association with the entity at hand -- and the identity testing and synonyms or alternate name forms can also play into this determination.

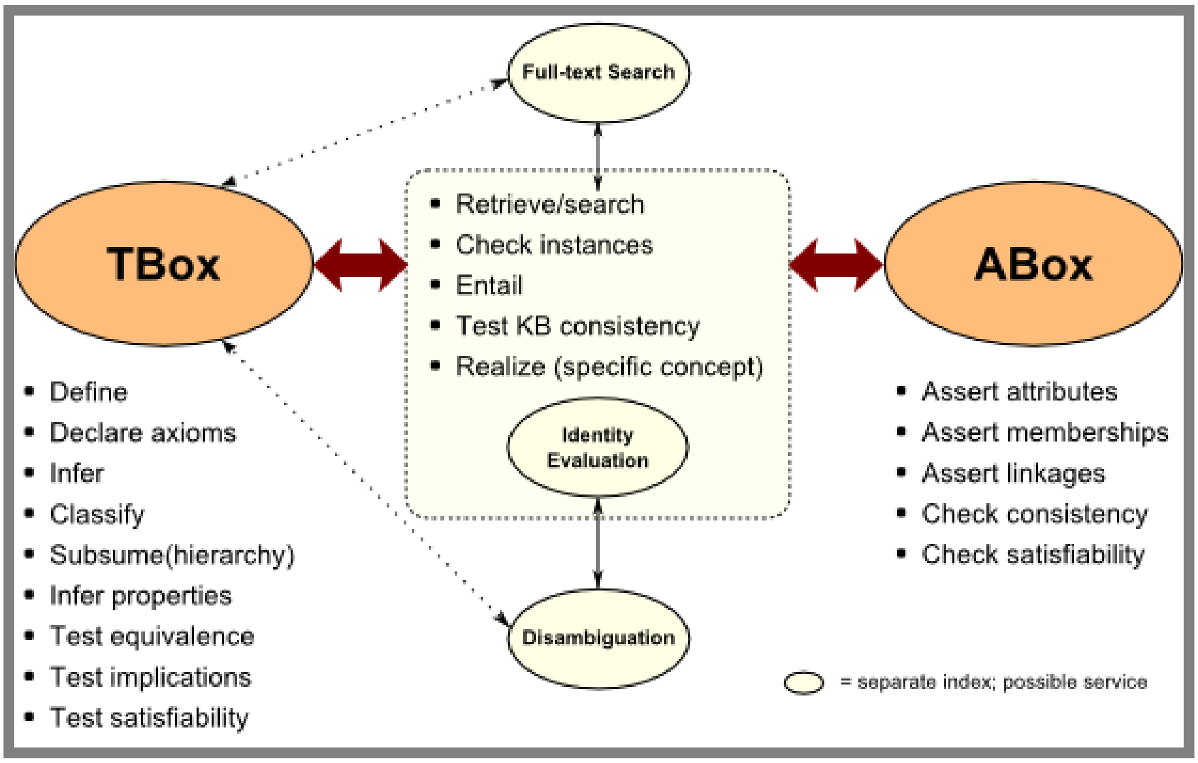
Again, this is ***work*** that has largely not been critical for the early phases of linked data, but is becoming essential as the application of linked data is being contemplated for doing actual, meaningful stuff.

No Reasoning or Inference

Of course, linked data and the ABox by definition don't provide this kind of work; reasoning and inference are appropriately the work of the TBox.

Some Implications for Architectural Design

We can now start diagramming these pieces into a conceptual architecture as follows:



This figure maps the ***work*** activities noted in the table at the top of this article and described throughout. Note there are a number of possible and specialized ***work*** activities at the interstices between the TBox and ABox, some of which are somewhat new to the description logics discourse as noted in the prior section.

I'm certainly no AI or KR expert, but it appears to me that there are pragmatic ***work*** tasks that emerge that do not easily fit into purely conceptual discussions of these things. The interstitial items possibly fall into this category, and those in the middle "bubbles" could even be done via separate processing or indexes not invoking the TBox level at all.

I suspect, though, that would be a mistake. The organizational view of the world at the TBox level provides useful reasoning and inferential bases for aiding other work tasks. For example, for disambiguation work, knowing that the instance of Joe Farmer was found in context with concepts relating to fertilizer or farm implements would help distinguish from Joe Farmer the lawyer (though would need additional clues or reasoning to dismiss that is was Joe Farmer the trucker shipping these things).

If you recall from Part 1, considerable discussion was devoted to the suggestion that linked data belonged in the ABox and that even if linkages are being made to other entities or instances, those assignments remain only assertions. We discussed identity evaluation and disambiguation in the previous section. These really come down to the questions of whether we are talking about the same or different things across multiple data sources.

Maintaining identity relations and disambiguation as separate components also has the advantage of enabling different methodologies or algorithms to be determined or swapped out as better methods become available. A low-fidelity service, for example, could be applied for quick or free uses, with more rigorous methods reserved for paid or batch mode analysis. Similarly, maintaining full-text search as a separate component means the ***work*** can be done by optimized search engines with built-in faceting (such as the excellent open-source Solr application). I will likely have more to say on this aspect after this series concludes.

A Couple Thoughts to Conclude this Part

People can do as they like and will within the semantic scope of the RDF and OWL languages. But, besides my recommendation to view linked data as ABoxes through the lens of description logics, I'd like to suggest a couple of additional 'best practices.'

I think linked data does itself a disservice to throw the `owl:sameAs` predicate around as much as it does. Ooooh! OWL; this is complicated stuff, no? And, actually, `owl:sameAs` is perhaps the most powerful entailment predicate around [5,6], which is also reflexive and transitive. How can we really use this property when all we are really talking about is an assertion at the ABox instance level?

I think use of predicates like this confuse ourselves and confuse the public. We give stuff a gloss and patina that is not supportable by any reasoning.

The OWL2 folks, I think, understood this issue in moving toward specific "assertion" language in the new draft version [7]. OWL2 now proposes these assertion predicates of `owl:SameIndividual`, `owl:DifferentIndividuals`, `owl:ClassAssertion`, `owl:ObjectPropertyAssertion`, `owl:NegativeObjectPropertyAssertion`, `owl:DataPropertyAssertion`, and `owl:NegativeDataPropertyAssertion`. Note that, axiomatically, the `owl:SameIndividual` is now the functional equivalent of the older `owl:sameAs` [8].

I think it is a step in the right direction, though frankly I would have preferred a semantics that used the "assert" naming for instances (individuals) as well.

But, actually, I have a more fundamental question, related to confusion argument above: Why should assertions be an OWL property? Would it not make better and cleaner sense to establish assertion predicates within RDF or RDFS and to minimize stronger implications of possible decidability or entailment? Would this approach not lead to a cleaner split between instance records (ABoxes) and linked data that could be kept solely within the RDF vocabulary without invoking OWL at all?

In this manner, we could leave the various emerging profiles for OWL2 [9] reserved for TBox-level reasoning. It would make for easier and understandable reasoners, while making it easier for publishers to expose linked instance data.

Well, OK; so, I have now gotten most of the arguments off my chest about ABoxes and linked data, the separation of concerns, and maintaining distinctions in language.

I will discuss in the next Part 3 what other researchers have to say on this use of the TBox and ABox and the split between them. Then, in the concluding Part 4, I discuss how this perspective might lead to a pretty simple RDF vocabulary and structure for linked data and instance records.

---

[1] This is our working definition for description logics:

"Description logics and their semantics traditionally split *concepts* and their relationships from the different treatment of *instances* and their attributes and roles, expressed as fact assertions. The concept split is known as the TBox (for *terminological* knowledge, the basis for *T* in *TBox*) and represents the schema or taxonomy of the domain at hand. The TBox is the structural and intensional component of conceptual relationships. The second split of instances is known as the ABox (for *assertions*, the basis for *A* in *ABox*) and describes the attributes of instances (and individuals), the roles between instances, and other assertions about instances regarding their class membership with the TBox concepts."

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2003. Sample chapters may be viewed from Enrico Franconi's Description Logics course notes and tutorial at http://www.inf.unibz.it/~franconi/dl/course/, which is an excellent starting reference point on the subject.

[3] This figure is from a presentation by Christian Bizer, "Linking Open Drug Data (HCLSIG LODD)," presented at the *ISWC 2008 Tutorial on Semantic Web for Health Care and Life Sciences*, Karlsruhe, Germany, October 27, 2008. See
http://carbon.videolectures.net/2008/active/iswc08_karlsruhe/prudhommeaux_swhcls/iswc08_prudhommeaux_swhcls_05.ppt. Also, for a broader evaluation of LODD datasets, see
http://esw.w3.org/topic/HCLSIG/LODD/Data/DataSetEvaluation.

[4] See, particularly, Chris Bizer, Richard Cyganiak and Tom Heath, "How to Publish Linked Data on the Web,", last updated July 2008, at http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/.

[5] `owl:sameAs` is even more powerful than `owl:equivalentClass`. Pat Hayes can often be counted on for clear explanations about semantic Web semantics, as this quote (hat tip to Tim Finin for pulling this out of the archives) describes:

"Allow me to suggest the appropriate intuition. In RDFS and OWL-Full there is a distinction between a class and the (set which is the) extension of the class. So two different classes might have the same sets of instances and yet still be distinct classes. (This is often described by saying that RDFS and OWLFull classes are 'intensional' as opposed to 'extensional'. For example, the classes of Human and HairlessBiped have the same members, but one might want to distinguish them since they have different defining conditions on membership. OWL-DL refuses to countenance such a possibility, although this may be rectified in OWL 1.1.) Thus there is an intuitive distinction in meaning between equivalentClass (having the same instances) and sameAs (being exactly the same thing). When, as in RDFS and OWL-Full, classes can have properties, one wants to preserve this distinction by saying that if A sameAs B then all the properties of A are also properties of B (since A and B are the very same thing); but this does not follow for A equivalentClass B, since A and B might still be distinct even if they do have the same members."

[6] Though long at 59 pp and geared to another purpose, I found first sections of this paper to also be very helpful in understanding entailments and decidability for the various semantic Web languages of RDF, RDFS and the dialects of OWL (Lite, DL and Full): Herman J. ter Horst, 2005. "Completeness, Decidability and Complexity of Entailment for RDF Schema and a Semantic Extension involving the OWL Vocabulary," paper preprint submitted to *Elsevier Science*, May 31, 2005, 59 pp. See http://www.websemanticsjournal.org/papers/20050719/document5.pdf.

[7] Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, eds., 2008. "OWL 2 Web Ontology Language:Structural Specification and Functional-Style Syntax," a *W3C Working Draft*, 02 December 2008. See for latest version http://www.w3.org/TR/owl2-syntax/. Note the XML serialization used is documented at http://www.w3.org/TR/owl2-xml-serialization/. I think it fair to say that many of these OWL2 changes were the result of mistakes and learning from the current OWL.

[8] Bernardo Cuenca Grau and Boris Motik, 2007. "OWL 1.1 Web Ontology Language Mapping to RDF Graphs," Editor's Draft May 23, 2007. See http://www.webont.com/owl/1.1/rdf_mapping.html.

[9] Boris Motik et al., eds., 2008. "OWL 2 Web Ontology Language: Profiles," a *W3C Working Draft*, December 2, 2008. See http://www.w3.org/TR/owl2-profiles/.