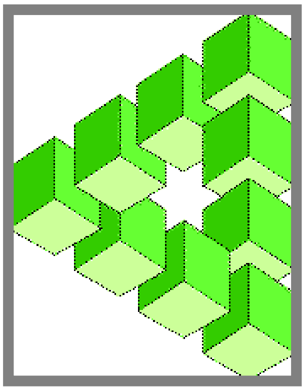
The Fundamental Importance of Keeping an ABox and TBox Split

by Mike Bergman - Sunday, May 17, 2009

http://www.mkbergman.com/489/ontology-best-practices-for-data-driven-applications-part-2/



Ontology Best Practices for Data-driven Applications: Part 2

It is perhaps not surprising that the first substantive post in this occasional series on ontology best practices for data-driven applications begins with the importance of keeping an ABox and TBox split. Structured Dynamics has been beating the tom-tom for quite a while on this topic. We reiterate and expand on this position in this post.

The Relation to Description Logics

Description logics (DL) are one of the key underpinnings to the semantic Web. DL are a logic semantics for knowledge representation (KR) systems based on first-order predicate logic (FOL). They are a kind of logical metalanguage that can help describe and determine (with various logic tests) the consistency, decidability and inferencing power of a given KR language. The semantic Web ontology languages, OWL Lite and OWL DL (which stands for description logics), are based on DL and were themselves outgrowths of earlier DL languages.

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. It is this construct for which Structure Dynamics generally reserves the term "ontology".

The second split of instances is known as the ABox (for *assertions*, the basis for *A* in *ABox*) and describes the attributes of instances (or individuals), the roles between instances, and other assertions about instances regarding their class membership with the TBox concepts. Both the TBox and ABox are consistent with set-theoretic principles.

Natural and Logical Work Splits

TBox and ABox logic operations differ and their purposes differ. TBox operations are based more on inferencing and tracing or verifying class memberships in the hierarchy (that is, the structural placement or relation of objects in the structure). ABox operations are more rule-based and govern fact checking, instance checking, consistency checking, and the like. ABox reasoning is generally more complex and at a larger scale than that for the TBox.

Early semantic Web systems tended to be very diligent about maintaining these 'box' distinctions of purpose, logic and treatment. One might argue, as Structured Dynamics does, that the usefulness and basis for these splits has been lost somewhat more recently.

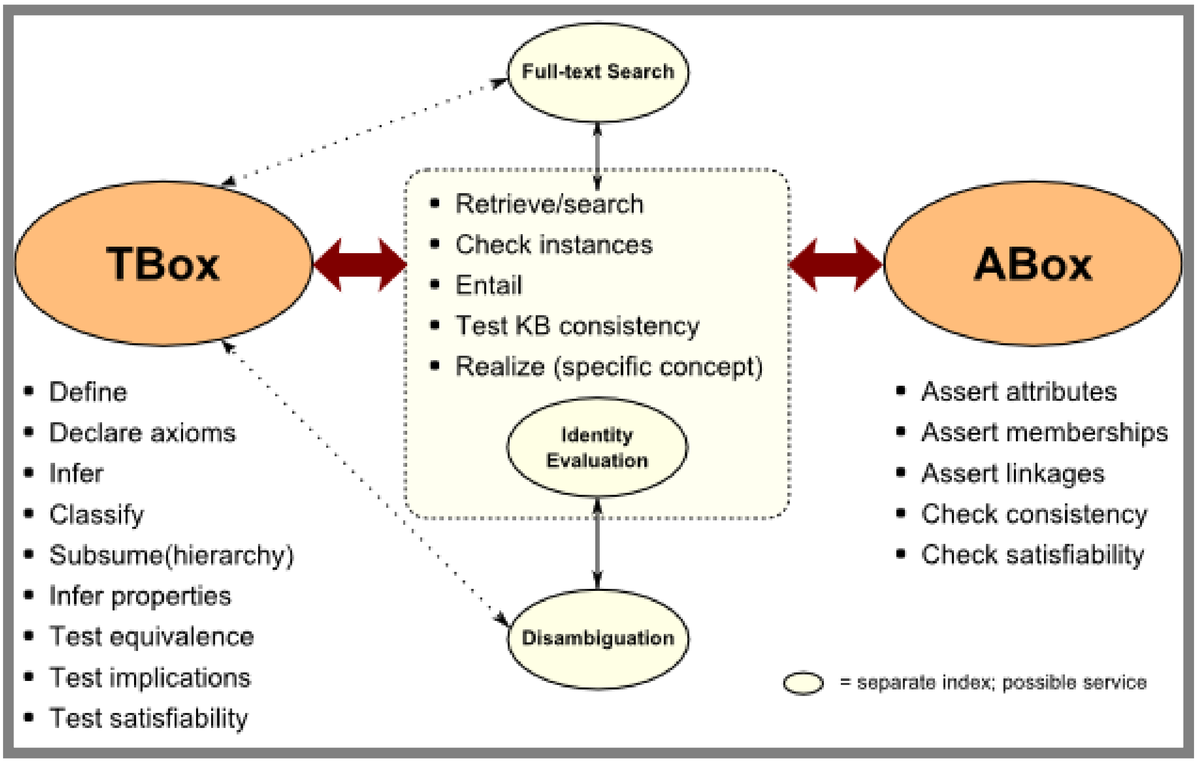
Particularly as we now see linked data become more prevalent, these same questions of scale and actual interoperability are posing real pragmatic challenges. To help aid this thinking, we have re-assembled, re-articulated and in some cases added to earlier discussions of the purposes of the TBox and ABox:

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)	• *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	• *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

- *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

- *Disambiguation*, which is resolving references to the proper instance

As the table 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. We can reflect these ***work*** splits via the following diagram:



This figure maps the ***work*** activities noted in the table, with particular emphasis on the possible and specialized ***work*** activities at the interstices between the TBox and ABox.

The Split Should Feel Natural

Whether a single database or the federation across many, we have data records (*structs* of instances) and a logical schema (*ontology* of concepts and relationships) by which we try to relate this information. This is a natural and meaningful split: structure and relationships *v.* the instances that populate that structure.

Stated this way, particularly for anyone with a relational database background, the split between schema and data is clear and obvious. While the relational data community has not always maintained this split, and the RDF, semantic Web and linked data communities have not often done so as well, this split makes eminent sense as a way to maintain a desirable separation of concerns.

The importance of description logics -- besides its role as a logical underpinning to the semantic Web enterprise -- is its ability to provide a perspective and framework for making these natural splits. Moreover, with some updated thinking, we can also establish a natural framework for guiding architecture and design. It is quite OK to also look to the interaction and triangulation of the ABox and TBox, as well as to specialized work that is not constrained to either.

For example, identity evaluation and disambiguation really come down to the questions of whether we are talking about the same or different things across multiple data sources. By analyzing these questions as separate components, we also gain 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).

These distinctions feel obvious and natural. They arise from a sound grounding in the split of the ABox and the TBox.

The Re-cap of Key Reasons to Maintain the TBox - ABox Split

So, to conclude this part in this occasional series, here are some of the key reasons to maintain a relative split between instances (the ABox) and the conceptual relationships that describe a world view for interpreting them (the TBox):

- We are able to handle instance data simply. The nature of instance "things" is comparatively constant and can be captured with easily understandable attribute-value pairs
- We can re-use these instance records in varied and multiple world views (the TBox). World views can be refined or approached from different perspectives without affecting instance data in the slightest
- We can approach data architectural decisions from the standpoints of the ***work*** to be done, leaving open special analysis or tasks like disambiguation or full-text search
- Ontologies (as defined by SD and focused on the TBox) are kept simpler and easier to understand. Inter-dataset relationships are asserted and testable in largely separate constructs, rather than admixed throughout
- Relatedly, we are thus able to use ontologies to focus on the issues of mappings and conceptual relationships
- Instance records can often be kept *in situ*, especially useful when incorporating the massive

amounts of data in existing relational databases
- Instance evaluations can be done separately from conceptual evaluations, which can help through triangulation in such tasks as disambiguation or entity identification
- It is easier to convert simple data structs to the instance record structure, aiding interoperability (a subject for a later part in this series)
- We provide a framework that is amenable to swapping in and out different analysis methods, and
- It is easier for broader input when the task is adding and refining attributes rather than internally consistent conceptual relationships.

Here is a final best practice suggestion when these ABox and TBox splits are maintained: Make sure as curators that new attributes added at the instance level are also added with their conceptual relationships at the TBox level. In this way, the knowledge base can be kept integral while we simultaneously foster a framework that eases the broadest scope of contributions.

This post is part of an occasional **AI3** series on ontology best practices.

PDF generated by *AI3:::Adaptive Information* blog