Cognescent

Technical Summary
August 2013

Product characteristics

Technically, the purpose of having built a framework for knowledge representation and processing (or Business Intelligence) is to leverage the contents of incoming business data (and metadata, if any available) and then converting this data into information for subsequent knowledge extraction. Data often comes from disparate sources, in disparate formats and usually in a form not suitable for integration and merge / exchange.

Here we mention three levels: data, information and knowledge as the layers in which each one serves as the basis for the upper. We aim to build a tool set which will enable go from the lower data elements level up to the knowledge layer with minimal intervention thus enabling loading and merge of huge datasets in a semi automated manner.

Going from one abstraction level up into another we attempt to alleviate manual intervention, infer types and resolve relationships. Abstraction over abstraction means we can extend what we could know by the data itself.

The ultimate abstraction of aggregate knowledge in our platform are 'Concepts' of given 'ConceptType'(s) inferred from the source data. Concepts can also play roles in 'State' 'Transitions' regarding the pattern its ConceptType resembles. All of this coming in an inter-source integrated and seamless manner to the end user.

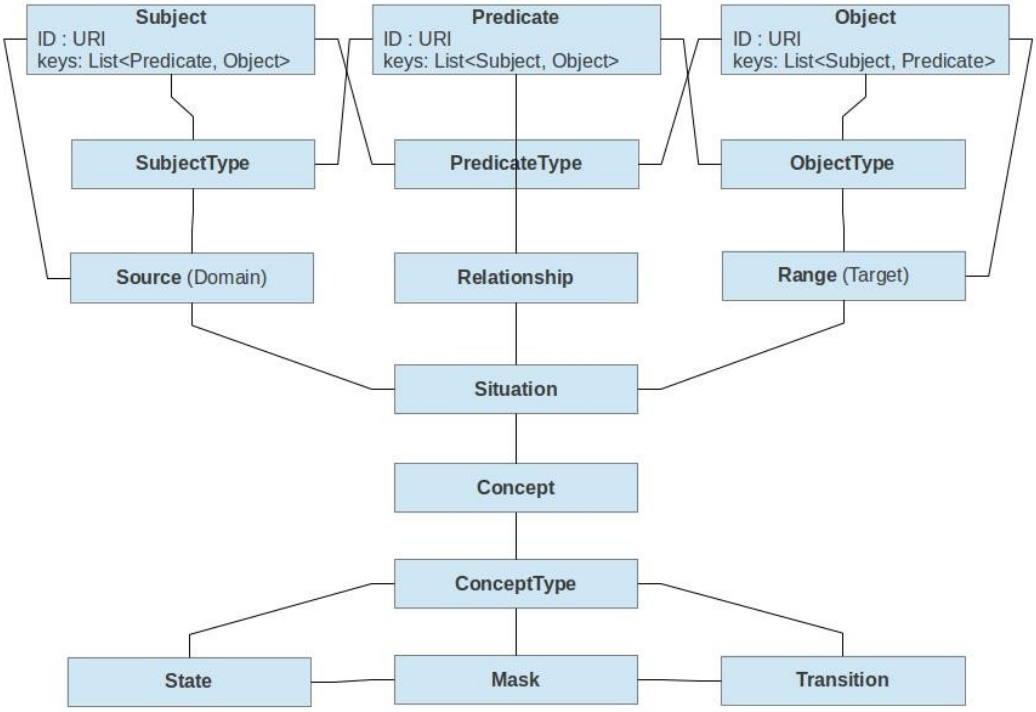
As a source data format (for loading and merging triples, aggregating relationship and structuring concepts) the approach of a kind of graph database, the RDF triple model, is the model we adopted because it seems to be the more appropriate mechanism of interaction with varying sources of data. Virtually any data source, relational database, NoSQL store, XML and others have straightforward mechanisms for converting into / from RDF.

Another important characteristic of our approach is, regardless there exists many mechanisms for specifying schema In any flavor of data representation, for the sake of the inferences we want to be capable of we don't want to rely on any such mechanism. Being tied to one such mechanism would signify that input data should be processed before converting it into triples and the resulting triples would have need to be processed too.

The diagram below shows how our model attempts to perform type inference at the triple level (data), then discovering relationships (information level) and finally aggregating Concepts (knowledge level).

Inferences desired over the model (without schema):

1. A Subject is of a given type (because it shares the same Predicates, same type)
2. An Object is of a given type (because of the Predicates who mention it / others)
3. A Predicate is of a given type (because of being functional between Subjects and Objects)
4. Domain, Range, Relationship (functional aggregation)
5. Situation (given previous Relationship, a Situation is an occurrence of a Concept)
6. Concept (aggregation of Situations corresponding to a Subject)
7. ConceptType (aggregation of 'schema' or Mask / properties of its Subjects)
8. State (aggregation of 'schema' or Mask / properties of its Subjects)
9. Transition (between two Concept State's, reified has entity on itself or as a rule)
10. Mask (query / declaration matches situation-triple components/parts).
11. Reification: Implicit for expanded relationships.
12. Reification: Discovery. Functional relationship composition.


Subject
ID : URI
keys: List<Predicate, Object>
Predicate
ID : URI
keys: List<Subject, Object>
Object
ID : URI
keys: List<Subject, Predicate>
SubjectType
PredicateType
ObjectType
Source (Domain)
Relationship
Range (Target)
Situation
Concept
ConceptType
State
Mask
Transition