Using Category Theory to facilitate multiple manufacturing service database integration

Ryan Wisnesky, Spencer Breiner, Albert Jones, David I. Spivak, Eswaran Subrahmanian

Ryan Wisnesky
Categorical Informatics
Cambridge, MA, USA
Email: ryan@catinf.com

Spencer Breiner
Software and Systems Division
Information Technology Laboratory
National Institute of Standards and Technology(NIST)
Gaithersburg, MD, USA
Email: spencer.breiner@nist.gov

Albert Jones
Systems Integration Division
Engineering Laboratory
National Institute of Standards and Technology(NIST)
Gaithersburg, MD, USA
Email:albert.jones@nist.gov

David I. Spivak
Department of Mathematics
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139
Email:dspivak@mit.edu

Eswaran Subrahmanian
Institute for Complex Engineered Systems and Engineering and Public Policy
Carnegie Mellon University
Pittsburgh, PA 15213
sub@cmu.edu

Corresponding Author: Eswaran Subrahmanian, Software and Systems Division, National institute of Standards and Technology, Mail Stop 8970, 100 Bureau Drive, Gaithersburg, MD, USA, Email: sub@cmu.edu

.

Using Category Theory to facilitate multiple manufacturing service portal integration

Abstract

The goal of this paper is to illustrate the use of category theory as a basis for the integration of supply chain databases. We begin by discussing existing work on using OWL ontologies to integrate supply chain databases. In this paper we use as our reference prior work by Kolvatunyu et.al (2013) on the use of OWL-based semantic web tools to study the integration of different manufacturing service capability (MSC) databases using a generic-model approach that they propose in their paper. We approach the same task using a different set of tools, specifically category theory and FQL, a functorial query language based on categorical mathematics. This work demonstrates the potential utility of category-theoretic information management tools, and illustrates some advantages of categorical techniques for the integration and evolution of databases. We conclude by making the case that category theoretical approach can provide a more flexible and robust approach to integration of existing and evolving information

Keywords: manufacturing supplier database, Category theory, ontology, semantic mediation, categorical informatics, information integration

1. Introduction

Historically, the suppliers of certain types of manufactured goods and services could be discovered in paper-based catalogues and registers. The advent of the internet changed the possibilities of discovering suppliers through broader, online, market mechanisms (Finger et.al, 1996). Today, a number of e-market portals promoting similar services have emerged. These portals provide an obvious advantage over earlier discovery mechanisms by enabling quick and easy access to suppliers using information contained in a variety of databases.

However, this advantage has been partially offset by the increasing difficulties in searching for specific manufacturing services across a plurality of databases. Such difficulties are exacerbated by the fact that most of these portals have their own proprietary database formats. This fact makes it difficult to merge these different databases into a single, accurate and consistent set of information. A further difficulty arises because each database has at least three different types of schemas – physical, logical, and external; and these types can vary across multiple databases. In addition, there can be classification types that include text entries, process descriptions, part catalogues, and equipment lists.

With such a broad range of classification types and database schemas used in supplier databases, finding a supplier with just the right capabilities to meet OEM requirements can be difficult. The technical genesis of this difficulty stems from the need to integrate information across the disparate classification types and their corresponding databases. Early efforts to address these integration problems were quite general and usually lacked sufficiently expressive semantics. As a result, they often omitted key information originally presented by the service providers, thereby failing to identify services precisely enough for matching them against OEM requirements.

2. The semantic mediation approaches

Recently, semantic-mediation techniques, based on ontologies, have been used to address this problem. The need for semantic mediation between different database schemas can arise 1) when requirements are based on one classification scheme but capabilities are based on another or 2) when there is a lack of taxonomic and hierarchical data regarding manufacturing services. In such cases, mediation approaches can be based on OWL (Bechhoffer, 2009, OWL, 2015), a family of Web Ontology Languages used to represent semantic knowledge. OWL is based on Description Logic (Krötzsch, et.al, 2012), a fragment of first-order logic. OWL is closely associated with a number of other web technologies including RDF, SWRL, and SPARQL. The Resource Description Framework (RDF) (RDF, 2013) is often used for data specification. Semantic Web Rule Language (SWRL) (Horrocks et al., 2004) is used for handling algebraic equations. And, the SPARQL Protocol and RDF Query Language (SPARQL) (Perez, et al., 2009) is used for querying and reasoning about ontologies.

Figure 1 goes here

3. Applying ontologies to the supply chain problem

As described by Kulvatunyou et.al in (2013), there are two ontological approaches to solving our supply chain problem: the generic-models approach and the reference-model approach. In both approaches, services and capabilities are classified using a variety of taxonomies, ranging from the materials involved in a process to the equipment used in that process. In some cases, these classifications involve substantial domain knowledge based in the particular industry of the service provider, such as aerospace or automotive. The generic-models approach translates each database into an ontology and maps these ontologies into one another. The reference-model approach constructs a common reference ontology, and develops mappings from each database ontology into this common one.

3.1 The Generic-model approach

Kulvatunyou et.al (2013) used a generic-model approach to develop models of manufacturing services and capabilities, as shown in Figure 1. In that same paper, the authors applied some of the aforementioned OWL-based semantic web tools to study the integration of different manufacturing service capability (MSC) databases using the generic-model approach. The authors also discuss the advantages and disadvantages of this approach. Of particular concern is that this approach requires the construction of 1-to-1 mappings between pairs of ontologies, and the number of such pairs scales quadratically with the number of databases; this is a problem if we hope to scale this approach to a large number of service providers. This issue can be mitigated, to some extent, by defining a few “atomic” mappings and constructing the rest by chaining them together, or by using the reference-model approach discussed below. A more significant obstacle arises when a database contains some unique information. That is, whenever provider A specifies some class of information unknown to provider B, there can be no 1-to-1 mapping of A’s ontology into B’s ontology. Worse, if every such provider specifies some unique type of data, there will be no 1-to-1 mappings at all!

3.2 The Reference-model approach

The reference approach addresses the problems with the generic-model approach by constructing a single over-arching reference ontology. One only needs to create a single mapping for each provider, locating the provider’s ontology inside the reference ontology. Furthermore, by providing sufficient detail in the reference ontology, one can ensure that it accepts 1-to-1 mappings from each provider’s ontology. However, the construction of the reference ontology itself is laborious and requires much more encoding of information than the generic model approach. But there is a long-term advantage to the reference ontology approach: adding a new portal requires only the mapping of the new portal ontology into the reference ontology. Of course, this advantage is contingent on the reference ontology being rich enough to accept the new portal; if it is not, then the reference ontology will also need to be updated to accommodate the new portal. Nonetheless, the reference-ontology approach is more useful for long-term maintenance. Additionally, it provides the flexibility needed to add new relationships in a centralized location rather than updating the mappings between individual ontologies.

Figure 2 Goes here

A second paper, Kulvatunyou, et al. (2014), forms the basis for our work. In this paper, the authors take two proprietary portals (called “A” and “B”) and integrate them using the reference ontology model illustrated in Figure 2. This integration enhances the ability to provide information about a manufacturing supplier by drawing on information contained in portal B to answer a query regarding entities from portal A. The authors begin with a relational database schema for each portal. Each schema is translated to an RDF model, which in turn is translated to an OWL model. Both are then embedded in a generalized “Upper Ontology” for manufacturing services developed by Ameri and Dutta (2006) using the Manufacturing Service Description Language (MSDL). The language and semantics of the reference ontology are independent of the two portal ontologies, rich enough to interpret them both, and, together, they serve as an interlingua between them. After mapping each of the portal ontologies into the reference ontology, the authors resolve linguistic differences between the two by mapping to terms encoded in the reference ontology.

Once this integration is accomplished, the result of a query to the integrated data will incorporate the data contained in both portals. Formally, the initial query is translated into two different queries, expressed in the syntax of the portals A and B. These are each instantiated in their respective OWL ontologies, which support the necessary reasoning about the two queries independently. The reference ontology then mediates and composes the individual results and returns them to the user. In effect, this approach provides the user with an integrated system containing enhanced information on each of the suppliers.

4. Limitations of ontology-based approaches

Because of their success in certain well-understood domains like manufacturing, ontologies have been quite popular. Nevertheless, ontology-based approaches to information integration have three major limitations. The first limitation is that expressing a domain-specific ontology using formal logic can be cumbersome and subject to numerous errors. This is especially true if the statements in the ontology form a single, monolithic structure in the domain's universe of discourse. The second limitation involves the construction of ontologies, which is a manual, time-consuming process. In particular, constructing an ontology becomes quite complicated when one must represent a large number of concepts and relationships. The third limitation involves using ontologies to facilitate information integration. A given domain may have several ontologies, each in some ways incomplete and/or ambiguous; moreover, these may be written in different ontology languages which implement different logical systems. It follows, then, that the ontology approach to information integration may require merging schemas, ontologies, languages and even logics (Goguen, 2004). This is the only way to ensure that semantics are respected throughout the entire "integration chain," from actual datasets and documents, through schemas and ontologies, all the way to logical frameworks.

One avenue to address such difficulties involves the creation of small, modular ontologies which can be combined as necessary to model particular domains. While a good idea in principle, significant problems arise in practice when the designer or user attempts to merge such ontological modules. One problem is the lack of a formal theory to provide a conceptual foundation for such integration. Another problem is the difficulty associated with extending and merging ontologies automatically and correctly

(Sowa, 2001, Perdiou et al., 2003). There are interactive tools for ontology merging and mapping, such as PROMPT, a component of the Description Logic-based tool Protégé (Noy and Musen, 2003). An evaluation of PROMPT has shown that there are limitations with respect to resolving schema-level conflicts (Kulvatunyou, 2011). As noted above, modularity can help, but only so far. The best alternative for reducing the complexity, time and number of errors involved in ontology construction would be a tool for automatic ontology creation. Indeed, some mechanisms have already been proposed and implemented for this task; for example, there are good tools for automatically creating noun-based classes. However, these usually fail to identify all relevant relationships between these classes.

In this paper, we address the same supplier database integration problems described above using a new approach to ontologies called "ologs" (Spivak and Kent, 2011). Ologs are a structured graphical representation of information based on a branch of mathematics called category theory, a discipline that studies structural properties independent from particular details of implementation. One of the major benefits of this new approach is that it eliminates the aforementioned problems.

5. What is Category Theory?

In the early 1940s, mathematicians began studying the subtle relationships and connections between different branches of mathematics. Many of the branches emerged from the set-theoretic revolution at the beginning of the 20th century, each with its own assumptions and terminology. While studying the relationship between two very different mathematical disciplines, concerned with geometric spaces on one hand and algebraic methods on the other, Saunders Mac Lane and Samuel Eilenberg discovered that they required a new language to describe their conclusions. That language is called *category theory* (CT) (Eilenberg & Mac Lane, 1945). Category theory enabled them to capture the relationships between these disciplines by focusing not on the objects of study themselves (spaces and algebras), which are very different, but rather on the relationships ("arrows") between those objects. The basis of these inter-object relationships is the structure-preserving function, which appears in both disciplines.

In general, the notion of a category is an abstraction of this idea of structure and structure-preservation. Categories can be compared using categorical constructions called functors, which translate the objects and relationships of one category into those of another. They do so in a precise way, preserving the integrity of these structure-preserving relationships. One can regard CT as the mathematical study of pure structure, in particular how to represent and compose such structure. As such, it can be useful in modeling the declarative semantics of a system, which are often distributed over many components organized into layers. The components of ''lower'' layers have very simple semantics, perhaps expressing first principles; higher layers are constructed as compositions of these lower-layer components. The latter may be supplemented with more complex semantics that have no analogue at the lower level, meaning that a higher layer can be more than the sum of its lower layers.

The most familiar example of a category is Sets, where the objects are sets, arrows are functions, and composition ("o") is defined by the familiar formula (g o f)(a) = g(f(a)). Other examples include the category of vector spaces with linear maps and the category of geometric spaces with continuous functions. One should regard an object in a category as a type of data, and an arrow as a process which converts one type of data into another. Composition determines the processes which arise from feeding the output of one process as the input of another.

5.1 Category Theory, formally

Formally, a category C contains two types of entities: objects and arrows. Objects are usually denoted with capital letters A,B,C,… while arrows use lowercase f,g,h, …. By definition, every arrow maps out of one object (its domain) and into another (its codomain); we represent this information by writing f:A→B. Most importantly, whenever two arrows match "tip-to-tail" C must provide a method for composing them, as in the diagram below:

$$\begin{tikzcd}
A \arrow[rr, "g\circ f"] \arrow[dr, "f"'] & & C \\
& B \arrow[ur, "g"'] &
\end{tikzcd}$$

The composition g∘f is read "g after f". The rules of categories require that composition is associative, meaning that if three arrows occur tip-to-tail, then triple composition h∘g∘f is the same no matter which order it is composed; i.e., there is no need to specify whether g and f are composed before h or vice versa. The final law of categories is that every object A has an identity arrow 1_A:A→A, which one might conceive of as a process which does nothing. The use of identity arrows will become clear below.
This definition of category is abstract (i.e., axiomatic). The advantage of this abstraction is its flexibility; nearly all structured information can be represented as a category, so many category-theoretic constructions and theorems will automatically apply to a wide range of examples. This points to a second advantage of categories: the language of objects and arrows is sufficiently expressive to define and study an impressive range of phenomena. We discuss a few examples below.

5.2 Isomorphism

One of the most fundamental notions in category theory is that of an isomorphism (or iso). An arrow f:A→B is invertible if there is another arrow g:A→B which composes with f to the identity: an isomorphism is an arrow which has an inverse, and two objects are isomorphic if there is some isomorphism which maps one to another. In Sets the isomorphisms are bijective functions, and two objects are isomorphic, written A≅B, exactly when they have the same number of elements.

$$g \circ f = 1_A \qquad \begin{tikzcd} A \arrow[r, bend left, "f"] & B \arrow[l, bend left, "g"] \end{tikzcd} \qquad f \circ g = 1_B$$

Even before the development of CT, mathematicians were aware that any useful mathematical statement that applies to one object also applies to any isomorphic object—indeed this property should define the notion of isomorphism: if two objects are isomorphic, they should be indistinguishable. As an example, consider the disjoint union A+B of two sets A and B. This is similar to the ordinary union of sets except that, if A and B share some objects, each of these appears twice inside A+B. To define the disjoint union in set theory one typically chooses two distinct tokens * and † and defines A+B as the set of pairs:

$$\{\, (a,*) \mid a \in A \,\} \cup \{\, (b,\dagger) \mid b \in B \,\}$$

The fact that * ≠ † ensures that shared objects in A and B are not identified in the disjoint union.

However, the above definition of disjoint union is somewhat arbitrary: we could have picked two different tokens, say ⋆ and ‡, and used these to define an alternative disjoint union A+^' B. Although the two notions of disjoint union are not exactly the same, they yield isomorphic results (namely, via the bijection that sends $(a,*) \mapsto (a,\star)$

and $(b, \dagger) \mapsto (b, \ddagger)$). Indeed, any two definitions of the disjoint union must yield isomorphic results, because the results must all have the same cardinality $|A| + |B|$..

The key insight is that disjoint union is a structural notion and, indeed, one can define disjoint union using nothing but the language of objects and arrows. When defined using category theory, it is a theorem that all constructions that define a disjoint union will yield isomorphic results, and it follows that disjoint unions must be invariant under isomorphism: if $A \cong A'$ and $B \cong B'$ then $A + B \cong A' + B'$. For more details, see Chapter 3 of (Spivak, 2014).

Now consider the ordinary set-theoretic union. This is not invariant under isomorphism, as in the following example, where $A \cong A'$and $B \cong B'$but $A \cup B \ncong A' \cup B'$.

$$A = \{a_1, a_2, a_3\} \cong \{a_4, a_5, c\} = A'$$

$$B = \{b_1, b_2\} \cong \{b_3, c\} = B'$$

$$A \cup B = \{a_1, a_2, a_3, b_1, b_2\} \ncong \{a_4, a_5, b_3, c\} = A' \cup B'$$

Does this indicate that union is not a structural construction? Not exactly. From a category-theoretic perspective, set-theoretic union smuggles in some structural information without explicitly declaring it, as we will see in the next section.

5.3 A Category-theoretic notion of union

One categorical definition of the usual notion of union is called a pushout. The pushout of two sets A and B is always computed relative to a specified overlap, O; the pushout is written A+_O B and O is the missing structural information in the naïve set-theoretic union. Consider the diagrams below, corresponding to the example in the previous section. Although their top rows are isomorphic, the overlaps are not; thus, once we explicitly include the overlap in our specification, these situations are no longer structurally identical.

$$\begin{tikzcd}
A & & B & A' & & B' \\
& \emptyset \arrow[ul] \arrow[ur] & & & \{c\} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

More generally, the pushout of two objects A and B is computed relative to a third object O, the overlap, and two arrows O→A and O→B which locate the overlap inside these objects. The pushout (i.e., the union) then forms the following diagram:

$$\begin{tikzcd}
& A \underset{O}{+} B & \\
A \arrow[ur] & & B \arrow[ul] \\
& O \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

The two upper maps locate A and B within the pushout, and the diagram commutes, meaning that the copies of O inside A and the copy of O inside B have the same location inside $A{+}_{O}B$.

Although we have focused here on constructions in **Sets**, it is important to remember that the structural definition of pushouts make sense in any category. Of particular importance in the rest of the paper is that we can interpret pushouts as operations on categories themselves.

5.4 More on functors

Because categories are mathematical objects they have their own notion of a structure-preserving function called a *functor*. A functor $F\colon \boldsymbol{C} \to \boldsymbol{D}$ sends objects of $\boldsymbol{C}$ to objects of $\boldsymbol{D}$ and arrows of $\boldsymbol{C}$ to arrows of $\boldsymbol{D}$, in such a way as to preserve domain and codomain, composition and identities. More explicitly, if $g: A \to B$ is an arrow in $\boldsymbol{C}$, then $F(g): F(A) \to F(B)$ is an arrow in $\boldsymbol{D}$**.** This means that composable arrows in $\boldsymbol{C}$ map to composable arrows in $\boldsymbol{D}$, and functors respect composition, as show in this diagram:

$$\begin{tikzcd}
\mathbf{C} \arrow[rrr, "F"] & & & \mathbf{D} \\
& A \arrow[dl, "g"'] \arrow[dd, "h \circ g"] & & & F(A) \arrow[dl, "F(g)"'] \arrow[dd, "F(h \circ g) \\ \| \\ F(h) \circ F(g)"] \\
B \arrow[dr, "h"'] & & & F(B) \arrow[dr, "F(h)"'] & \\
& C & & & F(C)
\end{tikzcd}$$

6. Ologs

Category theory provides a formal language to precisely specify aspects of systems that might usually be expressed in natural language or in ad hoc diagrams, enabling a formal understanding of a system's functionality, its components, or its computational behavior, for example. Ontology logs, or ologs, are a method for presenting a category in a human-readable format and, at the same time, specifying its intended semantics. Developed by Spivak and Kent (2010), an olog is a category in which the objects and arrows have been labeled by English-language phrases that indicate their intended meaning. The objects, which represent types of things, are labeled by noun phrases while the arrows, which represent functional relationships (also known as aspects, attributes, or observables), are labelled by verb phrases. In addition, "path equivalences" express facts about how these functional relationships interact.

Individual concepts and connections within one olog can be "functorially aligned" with concepts and connections in another. A functor creates a precise analogy between, for example, the work of one author to the work of another so that the precise nature of the comparison is explicitly specified rather than left to the reader's imagination. The ability to incorporate mathematical precision into the sharing of ideas is a central feature of ologs. The additional expressivity of ologs, relative to other graphical methods, provides semantic clarity and interoperability which cannot be achieved with graphs and networks. An additional advantage arises from the use of functors, which allow us to connect different ologs in a meaningful way, enabling sharing and "data fusion."

As explained in (Spivak & Kent), an olog is similar to a relational database schema and in fact, an olog can literally serve as a database schema if desired. The main advantages of creating an olog rather than writing a prose description of a subject are that:

- an olog gives a precise formulation of a conceptual worldview,
- an olog can be formulaically converted into a database schema,
- an olog can be extended as new information is obtained,
- an olog written by one author can be easily and precisely referenced by others,
- an olog can be input into a computer and "meaningfully stored", and
- different ologs can be compared by functors, which generate automatic
- translation systems.

Our vision of the future of data integration is one of many researchers building (relatively) small ologs associated with the type of work they do. When another researcher wants to use such a model (e.g., to exchange data), he or she would find the associated category and develop a functorial connection between that category and his or her own work. For example, (Spivak et.al, 2011) develops a formal relationship between a hierarchical protein material and a simple social network. In (Williamson et al., 2001) the authors apply category theory to formalize the underlying structure of knowledge embodied in certain industrial computation systems. Their goal was twofold: to produce software from a formalized knowledge representation that expresses its intended semantics, and to unify knowledge-based systems developed by many different organizations and written in a variety of languages. These examples show the extent to which category theory enables the elucidation of precise analogies between very different disciplines.

From a design perspective, one major benefit of ologs is that they bridge the gap between human readability and computer readability. Another benefit is that they give humans the ability to record ideas in a precise way without being experts in a computer language, while resting on a solid mathematical foundation. Ologs allows the incremental development of, and analysis of, ontologies by basing them in an interconnected hierarchy of theories. Category theory supplies operations on the hierarchy (e.g., pushout) which expresses the formation of complex theories from simple ones. Additional operations, not discussed here, correspond to the abstraction of shared concepts in an array of theories.

7. The olog Approach to manufacturing service portals integration

In section 2 we presented an integration problem involving overlapping portals for manufacturing service capabilities (MSC), and discussed previous work addressing this problem using OWL ontologies. Here we consider the same problem from a category-theoretic perspective, using ologs. In place of OWL we use the Functorial Query Language (FQL), a query language with underlying categorical foundations. The opensource FQL project can be found at http://categoricaldata.net, or see (Wisnesky and Spivak, 2014) for details.

As with the OWL approach, there are several strategies to achieve the desired information integration. The particular strategy we use in this paper is as follows:

- Translate each portal schema into a category in FQL.
- Identify the overlap between portals, represented as a diagram of categories.
- Construct a new category via pushouts, representing a unified ontology for both portals.
- Transfer existing data to the unified category via a "Kan extension."
- Query the unified category using FQL, or translate data back into a relational database.

For expediency we primarily focus on the expansion of portal A's data by a collection of synonym and hierarchy data about materials contained in an OWL ontology. This demonstrates the way that categories can provide a lingua franca for comparing and assimilating knowledge represented in different formalisms. The same methods apply to other integration problems, and we will discuss some considerations for other integration problems inline.

7.1 Importing relational and ontological data

As discussed in the previous section, an olog (i.e., a category) can be treated as a relational schema (specifically, a SQL schema), and FQL provides canonical translations from ologs to SQL schemas and vice-versa. Initially, portal A's data is presented as a relational (SQL) schema. Every table consists of an ID column (i.e., a primary key), a set of attribute columns containing strings or integers, and a set of foreign key columns that contain IDs from other tables. As discussed in (Spivak, 2014) this SQL schema can be regarded as the presentation of a category. The objects of this category are the tables of the schema, as well as the basic datatypes INT and STRING. Each column of a table corresponds to an arrow out of the table; foreign keys represent maps between tables while data columns correspond to maps into the objects INT and STRING. Call the resulting category PORT_A. An instance of the schema can then be represented as a functor $I: PORT_A \rightarrow \mathbf{Sets}$. Each object (i.e., table) corresponds to a set, specifically its set of rows. A datatype like INT maps to the actual set of integers. If we choose a column in the table and look at the value in a particular row, this determines a function. For example, foreign keys map rows in one table to rows in another. This is enough to fully determine the functor I.

FQL imports the SQL code defining a schema and an instance and translates them into corresponding code defining an FQL schema and instance. The original schema for Portal A, as visualized in Microsoft Access, is shown in Figure 3, and the translation of the data into FQL is shown in Figure 4.

Figure 3 and Figure 4 go here.

The ontology about synonyms and hierarchy information for materials, which we will call M, is written in OWL, and it was challenging to express in FQL because of the absence of translation tools between OWL and FQL. In the end, we reconstructed (part of) the ontology in FQL by hand. However, this is merely a matter of implementation: the interpretation of OWL in first-order logic (Krötzsch, et al., 2012), and of first-order logic in CT (Lambek & Scott, 1986) shows that such a translation could be automated.

Starting from the materials ontology developed by Ameri and Dutta (2009), the hand-coding began by identifying a set O consisting of relevant words from the ontology (e.g., “ferrous”) and a function $parent: O \rightarrow O$ which specifies a hierarchy among this vocabulary. For example, “metal” might be the parent of “copper”, and “material” the parent of “metal.”

Next we calculated the reflexive and transitive closure of this function, defining a relation $is\_a \subseteq O \times O$. Thus “a metal is a metal” (reflexivity) and “copper is a material” (transitivity). In this case, it was enough to compute the closure in three transitive steps. Call the resulting category Ont_M. Figure 5 (next page) shows the ontology and Figure 6 (next page) shows end result of the reflective and transitive closure of the parent function.

Figure 5 and figure 6 go here

7.2 Identifying the overlap

Conceptually, this is the primary step in our integration. Some of the words from portal A appear in the ontology while others do not. Some of the words that do not appear are nonetheless synonymous with some words that do appear.

Let N denote the set of words from portal A and, as above, O the set of words from the ontology. The overlap between them can be identified as a relation $syn \subseteq N \times O$ specifying when a word from portal A is synonymous with a word in the ontology. This set of pairs is the overlap between our two categories. Specifically, a set can be thought of as a discrete category which has no functional relations (except identities). This means that a functor out of a discrete category is completely determined by its effect on objects. Thinking of syn in this way determines a diagram of categories and functors as below:

$$\begin{tikzcd}
Port_A & & Ont_M \\
& syn \arrow[ul, "F"] \arrow[ur, "G"'] &
\end{tikzcd}$$

Since each word n∈N appears as an object of Port_A and each word o∈O as an object of Ont_M, the functors F and G are just the projection of each synonym pair (n,o) onto its first and second component.

7.3 Pushing out

The next step is to construct the pushout of this diagram in the category of categories. This operation has not yet been implemented in FQL, so this step had to be constructed by hand. Fortunately, the simple structure of the overlap category syn made this a relatively easy task. Intuitively, one glues together the two categories $\mathrm{Port}_{\mathrm{A}}$and $\mathrm{Ont}_{\mathrm{M}}$by setting equal any pair of objects related by syn.

$$\begin{tikzcd}
& Port_A \underset{syn}{+} Ont_M & \\
Port_A \arrow[ur] & & Ont_M \arrow[ul] \\
& syn \arrow[ul, "F"] \arrow[ur, "G"'] &
\end{tikzcd}$$

Ultimately, this determines a new relation $\mathrm{is\_a}' \subseteq \mathrm{N} \times \mathrm{N}$ where “n_1 is an n_2” exactly when there is a string of synonyms ($\sim$) and ontological relationships ($\subseteq$) like the one below:

$$\mathrm{n}_1 \sim \mathrm{o}_1 \subseteq \mathrm{o}_2 \sim \mathrm{n}' \sim \mathrm{o}_3 \subseteq \mathrm{o}_4 \sim \mathrm{n}_2.$$

The string above shows n_1 $\mathrm{is}'_{\mathrm{a}}\mathrm{of}$ n_2 because n_1 is a synonym of o_1 which is a synonym of n' which is a synonym of o_3 which is a subclass of o_4 which is a synonym n_2. Formally, this corresponds to the existence or non-existence of certain maps in the integrated category.

In some ways, this step is analogous to the reference ontology approach presented by Kulvatanyou, et al. (2013). Indeed, the pushout schema with its inclusion maps from Port_A and Ont_M is a reference ontology; it tells us how the concepts from the two domains interact. However, in contrast to the original reference approach, here the reference ontology can be computed automatically rather than specifically authored. Whereas the OWL approach necessitates the creation and maintenance of the entire reference ontology, our approach only requires identifying the overlap between two representations. Typically, this overlap will be much smaller than the entire reference ontology, saving us a good deal of work.

7.4 Transferring data

Because a database instance can be regarded as a functor from the database schema (represented as a category) into the category of sets and functions, a construction called a "Kan extension", written Σ, allows us to transport instances across functors between schemas:

$$\begin{tikzcd}
Port_A \underset{syn}{+} Ont_M \arrow[dr, dashed, "\Sigma(I)"] & \\
Port_A \arrow[u] \arrow[r, "I"'] & \mathbf{Set}
\end{tikzcd}$$

A full description of Kan extensions is beyond the scope of this paper, but see Chapter 7 of (Spivak, 2014) for details. Roughly speaking, the Σ-instance is constructed by first populating the tables from portal A and then closing this data under the functional relationships contained in the integrated schema; wherever data is missing, the instance is filled in with null values called Skolem variables.

7.5 Querying the integrated schema

We now have a category (schema) which extends the original portal A schema with additional information from the ontology. We also have an instance of this extended schema, populated from portal A's original data and the ontology. With these pieces in place we can query the resulting instance using terminology contained in either portal A or the ontology.

Even for queries involving only terminology from portal A, this new instance contains additional information not included in the original data from portal A. For example, the ontology contains information about many types of pre-hardened steel. A query regarding possible jobs on a particular drill type (which only invokes terms from portal A), may return jobs which involve these various steel types (see Figures 7 and 8).

Queries in FQL may be constructed in one of two ways. The first is via Kan extension, the same method used for transferring data, described above. Although this is a fully general method for specifying queries, the determination of which functors to extend along is often nontrivial. For this reason, FQL also implements a select/from/where (SFW) syntax analogous to the usual query mechanism for SQL. Alternatively, one can take the integrated FQL schema/data and translate it back into a relational database. This allows the application of all familiar database methods to study our integrated data

Figure 7 and Figure 8 go here

7.6 Integrating Multiple Portals

Now consider the full integration problem, which also involves a second portal B. Conceptually, the simplest way to integrate three or more models is to generalize from pushouts to colimits. Colimits, which involve essentially the same mathematical methods as pushouts, allow us to integrate models which fit together in more complex diagrams than those above. In the present situation, we would take a colimit over the following diagram, consisting of three schema categories and three overlap categories:

$$\begin{tikzcd}
 & \mathrm{colim}(A,B,M) & \\
Port_A \arrow[ur, bend left] & Ont_M \arrow[u] & Port_B \arrow[ul, bend right] \\
O_{AM} \arrow[u] \arrow[ur] & O_{AB} \arrow[ul] \arrow[ur] & O_{BM} \arrow[u] \arrow[ul]
\end{tikzcd}$$

Although mathematically justified, this naïve approach suffers from some pragmatic difficulties. Foremost among these is a scaling issue: we must identify the overlap between every pair of representations. Because the number of pairs scales quadratically, this means a lot of work when merging many representations. Equallydifficult, if we wish to add in a new representation to an existing merged model, we would need to specify its overlap with each of the original pieces.

Fortunately, we can get around these difficulties by building our integrated models in a step-wise fashion. (This step-wise approach also works when building a reference ontology using OWL; it is not specific to CT.) First pick two representations A and B, identify their overlap O and push out to form A+_O B. Then take a third representation M and identify its overlap O' with the integrated model A+_O B. This simultaneously represents the overlap of A with M and the overlap of B with M. Now push out to form a triple integration as in the diagram below:

$$\begin{tikzcd}
 & & & \left(Port_A \underset{O}{+} Port_B\right) \underset{O'}{+} Ont_M & & \\
 & Port_A \underset{O}{+} Port_B \arrow[urr] & & & & Ont_M \arrow[ull] \\
Port_A \arrow[ur] & & Port_B \arrow[ul] & O' \arrow[ull] \arrow[urr] & & \\
 & O \arrow[ul] \arrow[ur] & & & &
\end{tikzcd}$$

Obviously this process can be iterated to integrate any number of models. The procedure is linear in the number of models to be integrated; we need only identify one overlap for each. Moreover, it is naturally evolutionary: extending the integrated model by a new representation involves exactly the same process as the initial construction. Best of all, theorems about colimits guarantee that this iterated construction is correct—it always yields the same result as the naïve method above. We can even recover the partial overlaps using a construction called "pullback" which is a formal "dual" of pushing out.

8. Advantages of the olog approach

Based on our investigations, there are several advantages of the categorical approach to MSC integration, compared to the approach using OWL ontologies. The first advantage is the close relationship between categorical models and database specifications; this simplifies the use of categorical methods in real-world applications. In practice, this advantage manifests as a unique direct translation between database schemas and categories. In contrast, the translation from SQL to RDF to OWL may not be unique (and it is not direct), a fact which causes headaches when trying to rigorously prove the correctness of an integration algorithm. The close connection between categories and databases also allows us to leverage highly optimized database machinery to analyze our categorical models.

A second advantage is the existence of rigorous formal tools for working with these categorical representations. Pushouts yield an immediate definition for model integration. Even though this involves less work than the OWL approach (identifying overlap, rather than constructing an entire reference ontology), the results are more concrete. Whereas the OWL approach submits queries to a particular portal and then reconciles and mediates these queries through the reference ontology, we provide a tailored reference schema which can be queried directly. Similarly, Kan extensions allow us to automatically populate the integrated schema based on initial input data.

9. Conclusions

In this paper, we illustrated the use of category theory as an alternative approach to integration of two manufacturing supplier databases by contrasting it with the approach that uses a traditional ontology language, OWL. We also illustrated using our approach of identifying overlaps using synonyms between a given a portal and an ontology and using the push out operator to create a reference ontology. Further, by iteratively applying this approach a the portals to be integrated, we avoid the problem of quadratically increasing set of mappings that will be needed in other approaches. It also saves us the problem of creating and maintaining an entire reference ontology by allowing us just identify the overlap to evolve ontology incrementally. We have illustrated the advantages of the categorical approach to semantic integration of diversity of databases in the manufacturing service domain by contrasting with an OWL based approach. Our claim is that this approach is generalizable to other domains where integration of databases to provide services over the web.

Information integration depends on the interactions between the objects of study more than the internal workings of each individual object in isolation. Interaction is about structure, and 20th century mathematics has shown that structure is best understood through functional relationships. Therefore, to better understand integration, we must study structure and function. Structure and function are precisely the domain of category theory, which captures and describes the complex interdependencies between functional structures. Category theory allows us to describe information exchange and interdependency at the level of semantic meaning (process) rather than at the physical level of packets and symbols (a la Shannon). These capabilities make category theory a natural mathematical foundation for information integration.

Categorical foundations can enable a significant improvement in the way information is conceived, modeled and used. In particular, it allows for a diversity of formalisms to co-exist in a common context. These formalisms can be used both within and across disciplines, interfacing with data stores and interconnecting the programs and systems that rely on them. Category theory should facilitate the development of more open reasoning systems and the discovery of unexpected commonalities between information structures across disciplines, as it has already done in physics, computer science and mathematics. Finally, category theory can provide a precise formal foundation for emerging approaches to cross-disciplinary science, cloud-based services, systems engineering and cyber-physical and socio-technical systems.

The flexibility and semantic expressivity of the categorical data model as represented by ologs provides an opportunity for ologs to serve as a standard for data model exchange. Irrespective of the underlying model of data such as relational schema, first-order logic, or equational models, ologs may serve as a unifying model for comprehension and exchange. We are currently experimenting with the use of olog-based representations to manage data associated with the LAAMPS molecular dynamics simulation platform. Our goal is to connect various molecular dynamics tools through the construction of a unified underlying model, thereby capturing links among this data and facilitating chaining these simulations together. Once this experiment is done, we will evaluate the potential for using ologs as a standard for representing and exchanging data across simulation packages.

This type of interdisciplinary interaction is sorely needed in contemporary science, where each discipline has its own jargon and methods. Domain-specificity is useful because it allows the optimization of languages and tools to particular domain needs, but it also prevents information transfer between domains. In the best case, this leads to additional work (when the same results are developed independently in different places); in the worst case, important connections may go undiscovered and unexploited. The methods of category theory allow us to build bridges between these disciplines without disturbing local conditions. This hints at a relevant and useful mathematical foundation for the day-to-day business of engineers and scientists, where we turn tables of data into explicit categorical models. Surfacing the underlying semantics of these situations will expose them to easier sharing, reuse and unexpected analogies.

Acknowledgements

References

Ameri, F., and D. Dutta. 2006. “An Upper Ontology for Manufacturing Service Description.” In Paper Presented at the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, 10–13. Philadelphia: AIP Publishing.

Bechhofer, S. (2009). OWL: Web ontology language. In *Encyclopedia of Database Systems* (pp. 2008-2009). Springer US

Eilenberg, S. & Mac Lane, S. (1945) General Theory of Natural Equivalences. *Transactions of the American Mathematical Society*, 58:231-294.

Finger, Susan, M. Terk, E. Subrahamanian, Chris Kasabach, F. Prinz, Daniel P. Siewiorek, Asim Smailagic, John Stivoric, and L. Weiss. "Rapid design and manufacture of wearable computers." *Communications of the ACM* 39, no. 2 (1996): 63-70

Galison, P. (1997) *Image and logic: A material culture of microphysics*. University of Chicago Press, 1997.

Gougen, J, Data, Schemas and ontology Integration, International conference on Combined logic, 2004

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, *21*, 79

Kulvatunyou, B., Ivezic, N., & Lee, Y (2014): On enhancing communication of the manufacturing service capability information using reference ontology, International Journal of Computer Integrated Manufacturing, Online April, 2014.

Kulvatunyou, B., Ivezic, N., Lee, Y & Junho Shin , An analysis of OWL-based semantic mediation approaches to enhance manufacturing service capability models, International Journal of Computer Integrated Manufacturing, September 2013.

Krötzsch, M., Simancik, F., & Horrocks, I. (2012). A description logic primer. *arXiv preprint arXiv:1201.408*

Lambek, J. & Scott, P. (1986). *Introduction to Higher-Order Categorical Logic*. Cambridge University Press, Cambridge, UK.

Noy, N. F., & Musen, M. A. (2003). The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, *59*(6), 983-1024.

OWL, (2015), Ontology Web Language, http://www.w3.org/TR/owl-features/, May 1st, 2003 (accessed May 23, 2015)

Pérez, J., Arenas, M., & Gutierrez, C. (2009). Semantics and complexity of SPARQL. *ACM Transactions on Database Systems (TODS)*, *34*(3), 16.

Predoiu, L., Feier, C., Scharffe, F., de Bruijn, J., Martin-Recuerda, F., Manov, D., & Ehrig, M. (2005). D4. 2.2 State-of-the-art survey on Ontology Merging and Aligning V2. *EU-IST Integrated Project IST-2003-506826 SEKT*

Resource Description Language, 2013, http://www.w3.org/RDF/. (accessed, May 23rd, 2015)

Spivak and Wisnesky, (2014) Functorial Query Language Specification, http://categoricaldata.net/fql/fql_def.pdf

Spivak D. and Kent, R. ,(2011) Olog: a categorical framework for knowledge representation, PLOS, 2011. http://math.mit.edu/~dspivak/informatics/olog.pdf

Figure Captions

Figure 1. Generic model of manufacturing services capability (Kulvatunyou, et.al, 2013)

Figure 2. Reference ontology approach for integrating two MSC portals A and B

Figure 3. Schema for Portal A

Figure 4. Portal A data in FQL

Figure 5. Initial "IS_A" relationship from the ontology

Figure 6. Result of the "IS_A" relationship after transitive and reflexive closure

Figure 7. Query result on initial data displayed in FQL

Figure 8: Query result on enriched data displayed in FQL

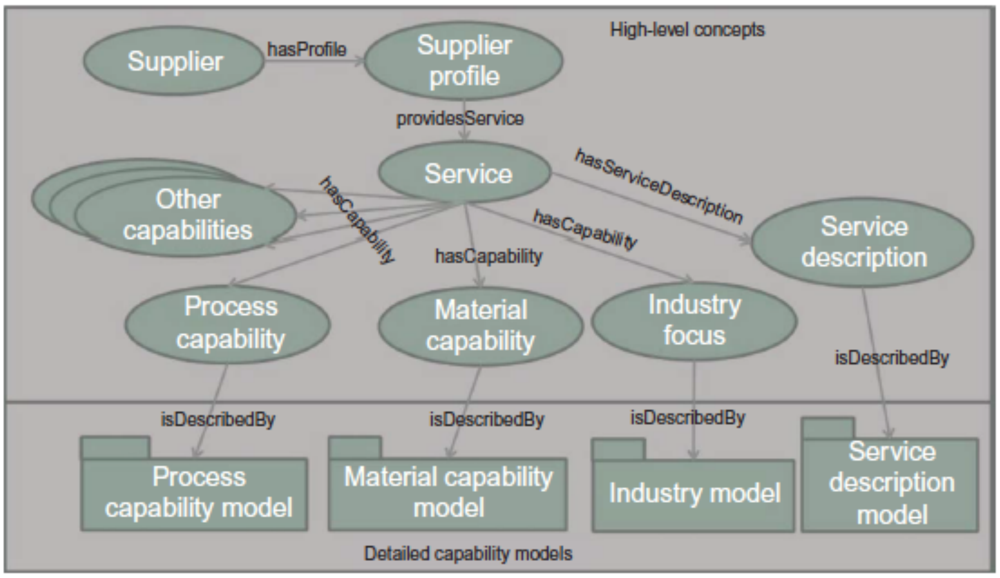


Figure 1. Generic model of manufacturing services capability (Kulvatunyou, et.al, 2013)

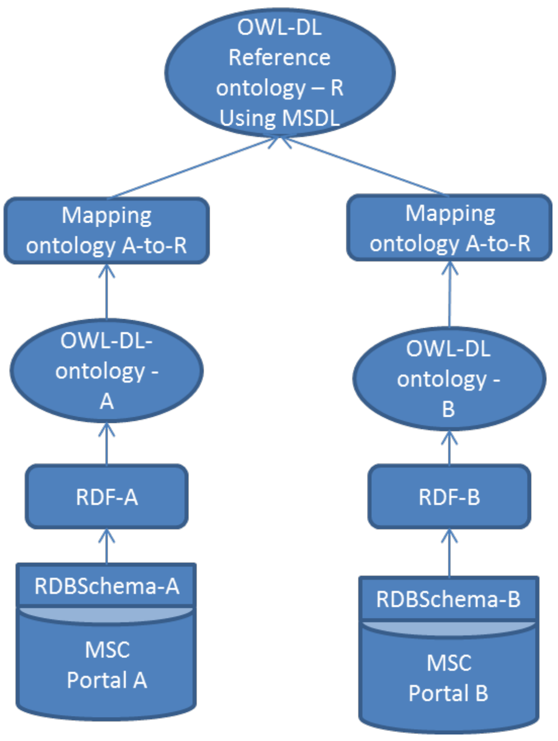


Figure 2. Reference ontology approach for integrating two MSC portals A and B.

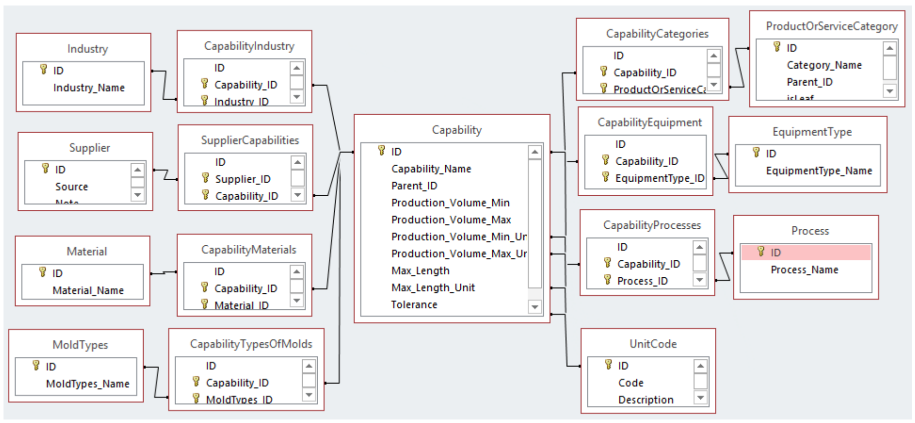
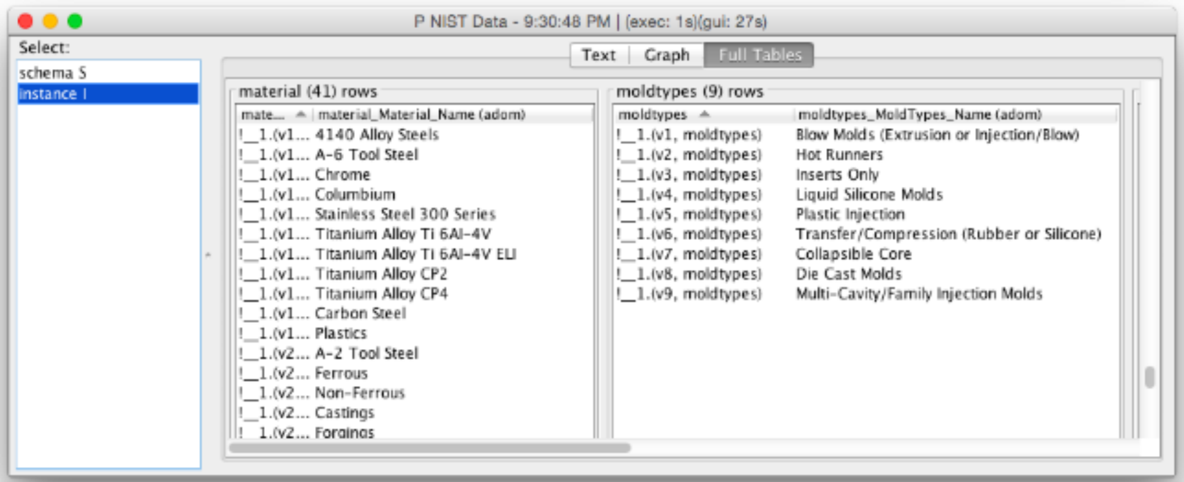


Figure 3. Schema for Portal A



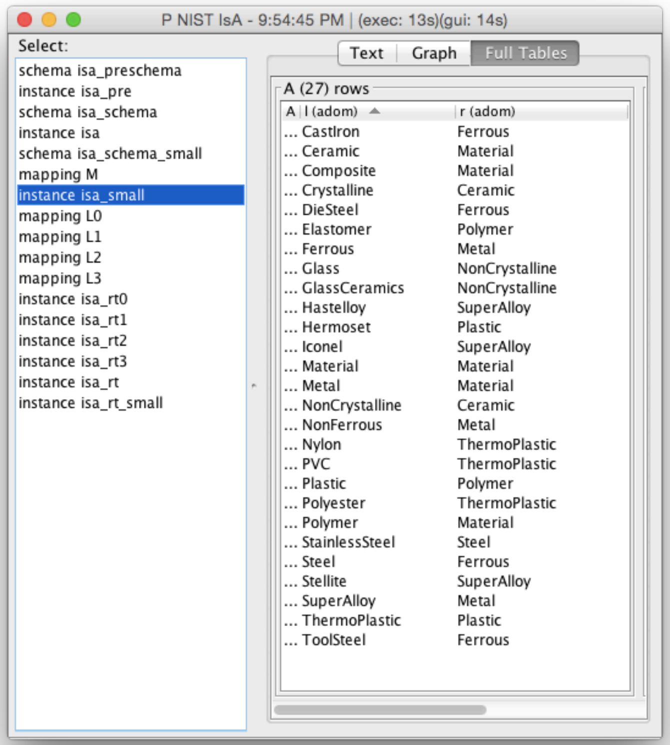
material (41) rows

mate...	material_Material_Name (adom)
!__1.(v1...	4140 Alloy Steels
!__1.(v1...	A-6 Tool Steel
!__1.(v1...	Chrome
!__1.(v1...	Columbium
!__1.(v1...	Stainless Steel 300 Series
!__1.(v1...	Titanium Alloy Ti 6Al-4V
!__1.(v1...	Titanium Alloy Ti 6Al-4V ELI
!__1.(v1...	Titanium Alloy CP2
!__1.(v1...	Titanium Alloy CP4
!__1.(v1...	Carbon Steel
!__1.(v1...	Plastics
!__1.(v2...	A-2 Tool Steel
!__1.(v2...	Ferrous
!__1.(v2...	Non-Ferrous
!__1.(v2...	Castings
!__1.(v2...	Forgings

moldtypes (9) rows

moldtypes	moldtypes_MoldTypes_Name (adom)
!__1.(v1, moldtypes)	Blow Molds (Extrusion or Injection/Blow)
!__1.(v2, moldtypes)	Hot Runners
!__1.(v3, moldtypes)	Inserts Only
!__1.(v4, moldtypes)	Liquid Silicone Molds
!__1.(v5, moldtypes)	Plastic Injection
!__1.(v6, moldtypes)	Transfer/Compression (Rubber or Silicone)
!__1.(v7, moldtypes)	Collapsible Core
!__1.(v8, moldtypes)	Die Cast Molds
!__1.(v9, moldtypes)	Multi-Cavity/Family Injection Molds

Figure 4. Portal A data in FQL



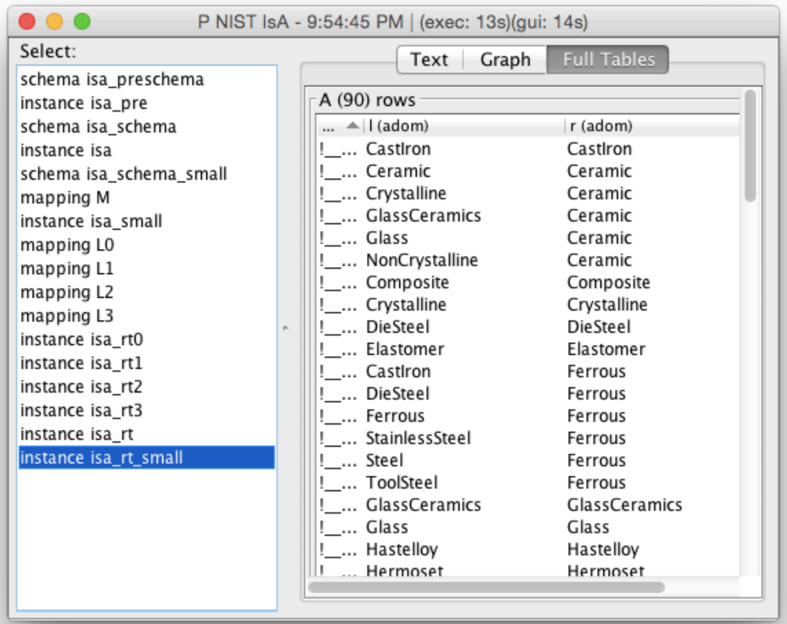
Select:

schema isa_preschema
instance isa_pre
schema isa_schema
instance isa
schema isa_schema_small
mapping M
instance isa_small
mapping L0
mapping L1
mapping L2
mapping L3
instance isa_rt0
instance isa_rt1
instance isa_rt2
instance isa_rt3
instance isa_rt
instance isa_rt_small

A (27) rows

A	l (adom)	r (adom)
...	CastIron	Ferrous
...	Ceramic	Material
...	Composite	Material
...	Crystalline	Ceramic
...	DieSteel	Ferrous
...	Elastomer	Polymer
...	Ferrous	Metal
...	Glass	NonCrystalline
...	GlassCeramics	NonCrystalline
...	Hastelloy	SuperAlloy
...	Hermoset	Plastic
...	Iconel	SuperAlloy
...	Material	Material
...	Metal	Material
...	NonCrystalline	Ceramic
...	NonFerrous	Metal
...	Nylon	ThermoPlastic
...	PVC	ThermoPlastic
...	Plastic	Polymer
...	Polyester	ThermoPlastic
...	Polymer	Material
...	StainlessSteel	Steel
...	Steel	Ferrous
...	Stellite	SuperAlloy
...	SuperAlloy	Metal
...	ThermoPlastic	Plastic
...	ToolSteel	Ferrous

Figure 5. Initial “IS_A” relationship from the ontology



P NIST IsA - 9:54:45 PM | (exec: 13s)(gui: 14s)

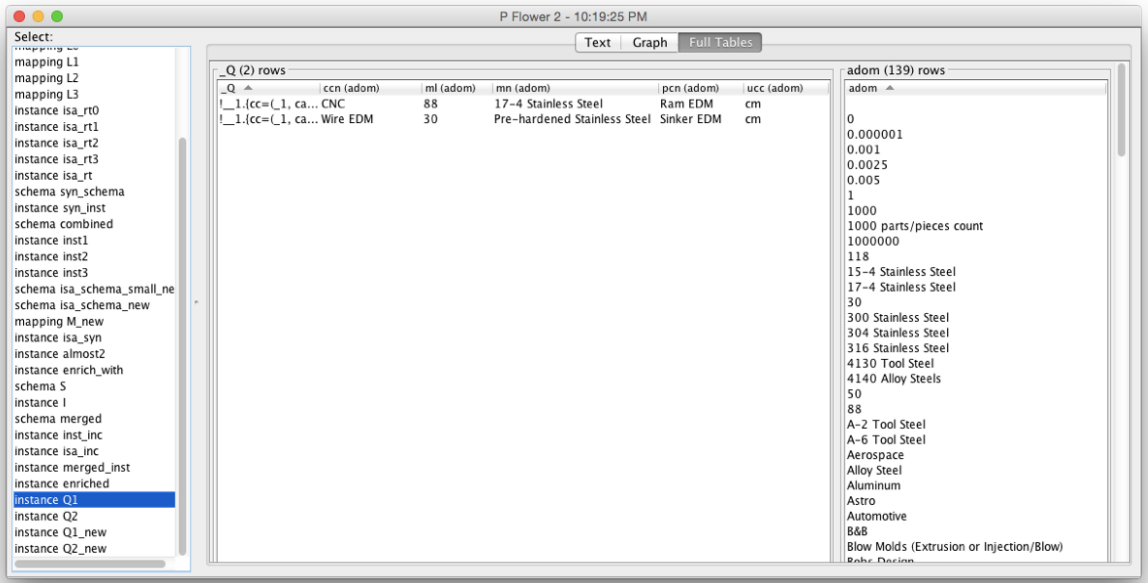
Select:

schema isa_preschema
instance isa_pre
schema isa_schema
instance isa
schema isa_schema_small
mapping M
instance isa_small
mapping L0
mapping L1
mapping L2
mapping L3
instance isa_rt0
instance isa_rt1
instance isa_rt2
instance isa_rt3
instance isa_rt
instance isa_rt_small

A (90) rows

...	l (adom)	r (adom)
!__...	CastIron	CastIron
!__...	Ceramic	Ceramic
!__...	Crystalline	Ceramic
!__...	GlassCeramics	Ceramic
!__...	Glass	Ceramic
!__...	NonCrystalline	Ceramic
!__...	Composite	Composite
!__...	Crystalline	Crystalline
!__...	DieSteel	DieSteel
!__...	Elastomer	Elastomer
!__...	CastIron	Ferrous
!__...	DieSteel	Ferrous
!__...	Ferrous	Ferrous
!__...	StainlessSteel	Ferrous
!__...	Steel	Ferrous
!__...	ToolSteel	Ferrous
!__...	GlassCeramics	GlassCeramics
!__...	Glass	Glass
!__...	Hastelloy	Hastelloy
!__...	Hermoset	Hermoset

Figure 6. Result of the "IS_A" relationship after transitive and reflexive closure



P Flower 2 - 10:19:25 PM

Select:

mapping L1
mapping L2
mapping L3
instance isa_rt0
instance isa_rt1
instance isa_rt2
instance isa_rt3
instance isa_rt
schema syn_schema
instance syn_inst
schema combined
instance inst1
instance inst2
instance inst3
schema isa_schema_small_ne
schema isa_schema_new
mapping M_new
instance isa_syn
instance almost2
instance enrich_with
schema S
instance I
schema merged
instance inst_inc
instance isa_inc
instance merged_inst
instance enriched
instance Q1
instance Q2
instance Q1_new
instance Q2_new

Text | Graph | Full Tables

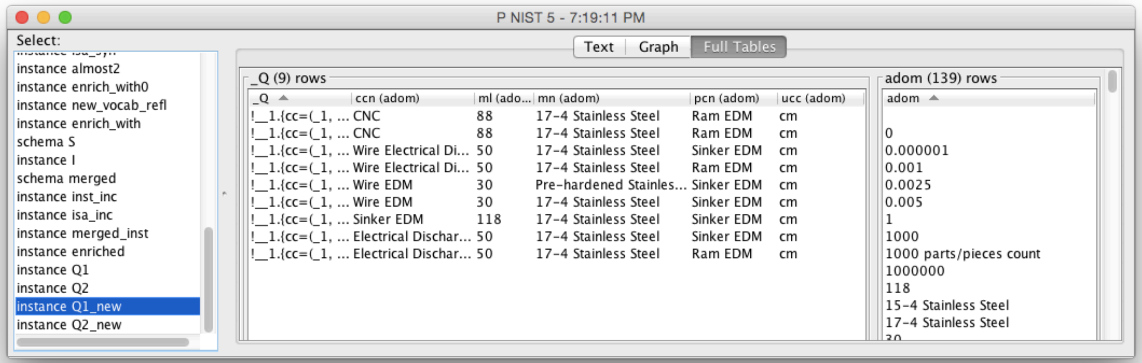
_Q (2) rows

_Q	ccn (adom)	ml (adom)	mn (adom)	pcn (adom)	ucc (adom)
!__1.{cc=(_1, ca...	CNC	88	17-4 Stainless Steel	Ram EDM	cm
!__1.{cc=(_1, ca...	Wire EDM	30	Pre-hardened Stainless Steel	Sinker EDM	cm

adom (139) rows

adom
0
0.000001
0.001
0.0025
0.005
1
1000
1000 parts/pieces count
1000000
118
15-4 Stainless Steel
17-4 Stainless Steel
30
300 Stainless Steel
304 Stainless Steel
316 Stainless Steel
4130 Tool Steel
4140 Alloy Steels
50
88
A-2 Tool Steel
A-6 Tool Steel
Aerospace
Alloy Steel
Aluminum
Astro
Automotive
B&B
Blow Molds (Extrusion or Injection/Blow)

Figure 7. Query result on initial data displayed in FQL



instance almost2
instance enrich_with0
instance new_vocab_refl
instance enrich_with
schema S
instance I
schema merged
instance inst_inc
instance isa_inc
instance merged_inst
instance enriched
instance Q1
instance Q2
instance Q1_new
instance Q2_new

_Q (9) rows

_Q	ccn (adom)	ml (ado...	mn (adom)	pcn (adom)	ucc (adom)
!__1.{cc=(_1, ...	CNC	88	17-4 Stainless Steel	Ram EDM	cm
!__1.{cc=(_1, ...	CNC	88	17-4 Stainless Steel	Ram EDM	cm
!__1.{cc=(_1, ...	Wire Electrical Di...	50	17-4 Stainless Steel	Sinker EDM	cm
!__1.{cc=(_1, ...	Wire Electrical Di...	50	17-4 Stainless Steel	Ram EDM	cm
!__1.{cc=(_1, ...	Wire EDM	30	Pre-hardened Stainles...	Sinker EDM	cm
!__1.{cc=(_1, ...	Wire EDM	30	17-4 Stainless Steel	Sinker EDM	cm
!__1.{cc=(_1, ...	Sinker EDM	118	17-4 Stainless Steel	Sinker EDM	cm
!__1.{cc=(_1, ...	Electrical Dischar...	50	17-4 Stainless Steel	Sinker EDM	cm
!__1.{cc=(_1, ...	Electrical Dischar...	50	17-4 Stainless Steel	Ram EDM	cm

adom (139) rows

adom
0
0.000001
0.001
0.0025
0.005
1
1000
1000 parts/pieces count
1000000
118
15-4 Stainless Steel
17-4 Stainless Steel

Figure 8: Query result on enriched data displayed in FQL