HHS Public Access

Author manuscript

Inf Syst Front. Author manuscript; available in PMC 2017 October 01.

Published in final edited form as:

Inf Syst Front. 2016 October ; 18(5): 953–965. doi:10.1007/s10796-016-9658-6.

Foundations of reusable and interoperable facet models using category theory

Daniel R. Harris[1]

Daniel R. Harris: daniel.harris@uky.edu

[1]Center for Clinical and Translational Sciences, University of Kentucky, Lexington, KY 40536, USA

Abstract

Faceted browsing has become ubiquitous with modern digital libraries and online search engines, yet the process is still difficult to abstractly model in a manner that supports the development of interoperable and reusable interfaces. We propose category theory as a theoretical foundation for faceted browsing and demonstrate how the interactive process can be mathematically abstracted. Existing efforts in facet modeling are based upon set theory, formal concept analysis, and light-weight ontologies, but in many regards, they are implementations of faceted browsing rather than a specification of the basic, underlying structures and interactions. We will demonstrate that category theory allows us to specify faceted objects and study the relationships and interactions within a faceted browsing system. Resulting implementations can then be constructed through a category-theoretic lens using these models, allowing abstract comparison and communication that naturally support interoperability and reuse.

Keywords

Data models; Interactive systems; Reusability; Information architecture

1 Introduction

Faceted browsing (also called faceted search or faceted navigation) is an exploratory search model, where facets assist in the navigation of search results (Hearst 2006a). Facets are simply attributes attached to the actual objects being explored. An example of a facet attached to a book could be its genre or publication date. In a typical faceted browsing system, a user is shown search results alongside a list of related, relevant facets, allowing interactive filtering and expansion of results (Hearst 2006b). A faceted taxonomy is the collection of facets provided by the interface and is often organized as sets, hierarchies, or graphs.

The design of a system's underlying model directly impacts the user's ability to filter, rank, and interact with the facets; in fact, some models contain no interactivity (Wei et al. 2013). Wei et al. observed three major theoretical foundations behind current research of facet models: set theory, formal concept analysis, and lightweight ontologies. Facet modeling

Correspondence to: Daniel R. Harris, daniel.harris@uky.edu.

focuses on the formal representation of faceted data and the interactive consequences that follow when using that model.

The motivation for choosing category theory began when designing the next phase of DELVE (Harris et al. 2014), our framework for creating visualizations for browsing biomedical literature. Specifically, we encountered difficultly in modeling DELVE's ability to create numerous visualizations, which are either controlled by facets or contain faceted structures. Additionally, one visualization may impact another (either by filtering or focusing). For example, how can one effectively represent a hierarchical tree of facets which is simultaneously browsable and capable of spawning faceted graphs containing interactive, linked nodes? A model of faceted browsing that is capable of representing faceted taxonomies generically would enable the quick creation of interoperable faceted components within an interface and enable their reuse in either other parts of the interface or in a different interface altogether. Although this abstraction is possible with set theory, the notation quickly becomes cluttered and error-prone. If set theory was used in the example above, individual sets would be necessary for each component and additional index sets would be needed for every type of relationship that maps a set to another set; linking nodes becomes an exercise in managing set indices and the sets necessary for facilitating interactivity. Additionally, it is difficult to incorporate existing work on faceted browsing due to the vast variety of models and implementations. A modeling methodology that is capable of operating at a high level of abstraction is necessary.

Some faceted systems, such as hierarchical faceted categories (Hearst 2006a; Yee et al. 2003), are implemented without a true theoretical foundation (Wei et al. 2013); in this context, categories refer to how facets categorically index items and is not related to category theory. In general, category theory aims to represent objects and relations at their most intrinsic, abstract level and is appropriate for modeling problems in the sciences (Spivak 2014), including computer science (Barr and Wells 1990). The volume of existing work for faceted browsing systems lends itself to a higher degree of abstraction, where existing works can become interoperable and reusable in new research settings. We will demonstrate that category theory is an appropriate framework for developing such abstractions by establishing facets and faceted taxonomies as categories in the mathematical sense.

2 Background

Faceted browsing systems enable effective use of faceted taxonomies and facet classification, the process of assigning facets to the resources to be queried. The utility of faceted classification and faceted taxonomies is well-understood (Hearst 2006a, b, Fagan 2013), even as a pivotal element to modern information retrieval (Dawson et al. 2006). Facet browsing is domain agnostic, but it is a natural fit for domains, such as biomedicine (Harris et al. 2014; Fuentes-Lorenzo et al. 2009), that have a rich history of ontological integration. Faceted taxonomies can aid in the construction of information models (Chu and Chow 2010) or aid in the construction of a larger ontology (Prieto-Díaz 2003). We focus on modeling faceted browsing in a way that enables the design of reusable and interoperable faceted taxonomies within the interface.

In order to fully understand our motivation, we must discuss existing faceted browsing models. When discussing these efforts, it is important to keep in mind that these models were constructed for a single system with a single faceted taxonomy; although each system was clearly an innovative and successful initiative, reusability and interoperability with future systems was not a priority or consideration discussed.

Additionally, basic knowledge of category theory is necessary to understand our model. We will introduce category theory in Section 2.2 using Spivak's definition of categories which was originally aimed toward the sciences audience (Spivak 2014).

2.1 Foundations of facet models

Of the three major foundations of facet modeling, set theory efforts tend to provide a model that explicitly includes interactive operations, such as filtering and ranking (Wei et al. 2013). Formal concept analysis focuses on defining facets and faceted structures for knowledge (Priss 2000) and has deep-seated roots in lattice theory in order to provide an organizational structure to faceted browsing. Lightweight ontologies provide an easy way to apply natural-language labels to concepts organized in an ontology (Giunchiglia et al. 2007), but do not explicitly model interactive operations.

Any implementation of faceted browsing, whether its foundation be grounded in set theory or lattice theory, could be abstracted into a category-theoretic framework as objects and relations. Because natural connections between category and set theory exist (Blass 1984; Spivak 2014), our work is most comparable to existing efforts in set theory. The facet modeling efforts that explicitly use set theory as a foundation differ in their core definitions and how they model filtering and ranking of facet objects: Dynamic Taxonomies (Sacco and Tzitzikas 2009) is a classic way of dynamically representing taxonomies with *is-a* relationships; Category Hierarchies are defined as connected, rooted directed acyclic graphs (Li et al. 2010); Generalized Formal Models (Clarkson et al. 2009) use entity-relationship diagrams to represent faceted hierarchies; FaSet (Bonino et al. 2009) implements facets and queries as sets within relational databases. Each of these implementations has its own base definition of what it means to be a facet. In FaSet, a facet F is a set of items and if the system has multiple facets, they are disjoint: $F_a \cap F_b = \varnothing$ (Bonino et al. 2009). The model is then constructed axiomatically using the base definition of a facet.

Our work is also similar to efforts based on formal concept analysis (Priss 2000) simply due to their shared abstract nature. We focus on modeling faceted structures once a representation for knowledge has been chosen, including lattices from formal concept analysis; we do not compete with formal concept analysis, but rather enable its reuse by providing a model capable of representing it consistently with other faceted structures. In other words, a lattice can peacefully coexist and interact with simpler structures such as sets and hierarchies. Many systems contain only one faceted taxonomy, but systems like DELVE can contain multiple visualizations; these visualizations either contain or are controlled by independent faceted taxonomies that may or may not share the same structure.

An example would be an interface that initially contains a visualization of a simple dynamic hierarchy depicting basic *is-a* relationships; for a given node, the interface could

interactively allow one to visualize more complex relationships that are stored in a different knowledge structure, such as graphs or lattices.

Another example would be augmenting a faceted taxonomy with additional facets from an external faceted taxonomy. A more concrete example will help illustrate how such a situation can arise naturally. At our local university hospital, our data warehouse provides a faceted interface which allows individuals to obtain aggregate counts of patients who match facets selected by the user; this allows quick feasibility checks of clinical research projects. Facets are arranged as a simple hierarchy and include items such as demographics (age, race, marital status) and vital signs (height, weight, body mass index, blood pressure, heart rate, respiratory rate). These facets, illustrated in Fig. 1, are based on what the electronic medical records (EMRs) for our university hospital provides. In Fig. 1, lines represent relationships and ellipses imply there are some relationships not shown for space considerations.

The EMR also uses drug codes that link to an external proprietary system where drugs are organized as a two-level hierarchy. For example, as seen in Fig. 2, buprenorphine *is-an* analgesic and an analgesic *is-a* central nervous system agent. We include this external hierarchy as part of our faceted taxonomy by adding drug as a facet. This augmentation enables one to search for classes of drugs, rather than just the drug codes found directly in the patient's EMR.

2.2 Category theory

Category theory has been demonstrated to be practical and useful for modeling problems in the sciences (Spivak 2014), including physics (Coecke and Paquette 2011), cognitive science (Phillips and Wilson 2010), and computational biology (Spivak et al. 2011). In database theory, categories can model databases (Spivak 2009, 2014) and can elegantly support data migration between schemas (Spivak 2012). Additionally, ologs use category theory for representing knowledge and modeling real-world situations with the goal of enabling reusable, transferable, and comparable research (Spivak and Kent 2012). Category theory can also be used in conjunction with semiotics as a foundation for information visualization (Vickers et al. 2013), where the process of visualization forms a category.

Because our model leverages known categories, a working knowledge of category theory is necessary. Informally, a category $\mathscr{C}$ is defined by stating a few facts about the proposed category (specifying its objects, morphisms, identities, and compositions) and demonstrating that they obey identity and associativity laws.

Definition 1 A category $\mathscr{C}$ consists of the following (Spivak 2014):

1. A collection of objects, $Ob(\mathscr{C})$.
2. A collection of morphisms (also called arrows). For every pair $x, y \in Ob(\mathscr{C})$, there exists a set $Hom_{\mathscr{C}}(x, y)$ that contains morphisms from x to y (Spivak 2014); a morphism $f \in Hom_{\mathscr{C}}(x, y)$ is of the form $f: x \rightarrow y$, where x is the domain and y is the codomain of f.
3. For every object $x \in Ob(\mathscr{C})$, the identity morphism, $id_x \in Hom_{\mathscr{C}}(x, x)$, exists.

4. For $x, y, z \in Ob(\mathscr{C})$, the composition function is defined as follows: $\bigcirc : Hom_{\mathscr{C}}(y, z) \times Hom_{\mathscr{C}}(x, y) \rightarrow Hom_{\mathscr{C}}(x, z)$.

Given 1-4, the following laws hold:

1. identity: for every $x, y \in Ob(\mathscr{C})$ and every morphism $f: x \rightarrow y$, $f \bigcirc id_x = f$ and $id_y \bigcirc f = f$.
2. associativity: if $w, x, y, z \in Ob(\mathscr{C})$ and $f: w \rightarrow x$, $g: x \rightarrow y$, $h: y \rightarrow z$, then $(h \bigcirc g) \bigcirc f = h \bigcirc (g \bigcirc f) \in Hom_{\mathscr{C}}(w, z)$.

As an example, **Set** is the category whose objects are sets and whose morphisms are functions between sets (Barr and Wells 1990; Spivak 2014). This implies that the set theoretic implementations of faceted browsing could also be directly abstracted and argued through category theory; we can comment on how our model could consume other models after we outline how our model works. The point of such structures is that they generalize sets by specifying families of elements rather than single elements, a formalization which enables exploration of structural similarities (Spivak 2014).

An easy way to construct a new category is to modify an existing category and create a subcategory by taking a subset of its objects and morphisms. One can think of this as simply removing some of the objects from the original category, then removing any morphisms that have lost their domain or codomain. The two categories behave similarly for those objects and morphisms found in both: they have the same identities and the same composition formula (Barr and Wells 1990).

Definition 2 A category $\mathscr{B}$ is a subcategory (Barr and Wells 1990) of $\mathscr{C}$ if

1. $Ob(\mathscr{B}) \subseteq Ob(\mathscr{C})$
2. for all $x, y \in Ob(\mathscr{B})$, $Hom_{\mathscr{B}}(x, y) \subseteq Hom_{\mathscr{C}}(x, y)$
3. the identity morphisms and compositions for $\mathscr{B}$ are copied from $\mathscr{C}$.

In our model, we will relate the concept of a facet and facet taxonomy to existing, well-known categories: **Rel** and **Cat**.

Definition 3 Rel is the category of sets as objects and relations as morphisms (Barr and Wells 1990), where we define relation arrows $f: X \rightarrow Y \in Hom_{\mathbf{Rel}}(X, Y)$ to be a subset of $X \times Y$.

In other words, any subset of $X \times Y$ is a relation from X to Y. Any binary relation is allowed, but most examples demonstrate the utility of "<", "≤", and "⊆". This category uses the composition of relations instead of functional composition; if $f \in Hom_{\mathbf{Rel}}(X, Y)$ and $g \in Hom_{\mathbf{Rel}}(Y, Z)$, then $(x, y) \in g \bigcirc f$ if and only if for some $y \in Y$, $(x, y) \in f$ and $(y, z) \in g$. The identity morphisms, $id_X \in Hom_{\mathbf{Rel}}(X, X)$, are the so-called diagonal relationships $\{(x, x) | x \in X\}$. **Set** is actually a subcategory of **Rel** (Barr and Wells 1990).

Definition 4 Cat is the category of categories. The objects of **Cat** are categories and the morphisms are functors (mappings between categories).

We informally defined functors as mappings between categories, but additional conditions are needed.

Definition 5 A functor F from category $\mathscr{C}_1$ to $\mathscr{C}_2$ is denoted $F: \mathscr{C}_1 \rightarrow \mathscr{C}_2$, where $F: Ob(\mathscr{C}_1) \rightarrow Ob(\mathscr{C}_2)$ and for every $x, y \in Ob(\mathscr{C}_1)$, $F: Hom_{\mathscr{C}_1}(x, y) \rightarrow Hom_{\mathscr{C}_2}(F(x), F(y))$. Additionally, the following must be preserved:

1. identity: for any object $x \in Ob(\mathscr{C}_1)$, $F(id_{\mathscr{C}_1}) = id_{F(\mathscr{C}_1)}$.
2. composition: for any $x, y, z \in Ob(\mathscr{C}_1)$ with $f: x \rightarrow y$ and $g: y \rightarrow z$, then $F(g \circ f) = F(g) \circ F(f)$.

Functors also play an important role in constructing the underlying graph of a category, which will be a key element of creating reusable facets and faceted structures.

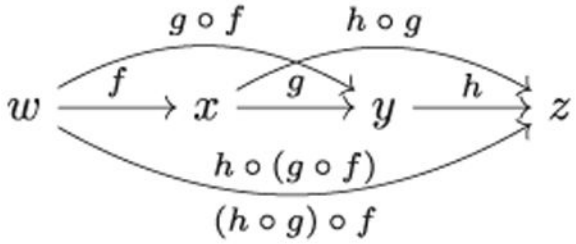
2.3 Why category theory?

We mentioned that category theory has been used to model several other practical problems, but we should comment on why it is also appropriate to model faceted browsing. The core issue is that faceted taxonomies come in many shapes and forms; this heterogeneity, while a sign that new and novel systems are being developed, is counterproductive for reusing faceted information effectively. Category theory provides the language for reasoning with these diverse structures in a consistent and productive environment. Once a category is defined, it becomes a bed for computations, such as transformations and products. As an example, consider n-ary products of objects within a category (Barr and Wells 1990), which act as a categorical version of n-ary Cartesian products.

Definition 6 The n-ary product of a list $A_1, A_2, \ldots, A_n$ containing n objects (not necessarily distinct) of a category is an object A with morphisms $p_i: A \rightarrow A_i$ for $i = 1,\ldots, n$. This is denoted as $A_1 \times A_2 \times \ldots \times A_n$ or simply $\prod_{i=1}^{n} \mathbf{A}_i$ Barr and Wells 1990).

We will demonstrate later that n-ary products model faceted universes and faceted queries. Beyond products, category theory is conceptually consistent with faceted browsing in that relations within a faceted taxonomy naturally mimic the constraints of categories. Consider a faceted taxonomy with *is-a* relations and observe that the following analogies hold:

1. An object has an identity function: x *is-a* x.
2. Relations can be composed: (x *is-a* y) *is-a* z.
3. Commutative diagrams typically demonstrate how objects and morphisms in a category obey associativity:



The same works for *is-a*-relations: ((*x is-a y*) *is-a z*) is equivalent to (*x is-a* (*y is-a z*)).

Given this discussion, there are clear benefits in choosing category theory as a modeling foundation, but we must acknowledge that the learning curve can be an impediment of adoption by those less familiar with the theory. We argue that any unfamiliar theory can be difficult to learn and that the benefits outweigh the obstacles.

3 A category-theoretic model

A faceted system is comprised of many implicitly intertwined parts: facets, a taxonomy that organizes the facets, an ability to select or focus on certain facets, and an ability to present the results of a selection or faceted query in an effective manner (Wei et al. 2013). Each of these components and their extensions can be abstractly modeled with category theory.

3.1 Taxonomies

We wish to be as general as possible in our abstractions so that any system with any faceted taxonomy can be modeled, regardless of the particular nuances of the facets and intra-facet relationships. There are two natural questions: how does one represent any taxonomy using category theory and how does one take that representation and augment with additional structure that supports the concepts of facets and faceted browsing? In Section 2.2, we introduced **Rel**, the category of relations, and we can restrict the morphisms of **Rel** to only correspond to the ⊆ relations:

Definition 7 Let **Tax** be a sub-category of **Rel**, the category of sets as objects and relations as morphisms where *Ob* (***Tax***) = *Ob*(**Rel**) and let the morphisms be the relations that correspond only to the ⊆ relations. The identity and composition definitions are simply copied from **Rel**.

The other relations that could possibly be represented by **Rel**, such as "<" and " ≤", are meaningless for categorically organizing concepts into a taxonomy. In other words, **Tax** is just a slimmer version of **Rel**, where we know exactly what binary relation is being used to order the objects. By itself, **Tax** is not particular helpful for modeling facet browsing: faceted browsing is piloted by interacting with disjoint collections of facets, which we will refer to as facet types. This will allow us to apply additional structure and granularity needed to support faceted browsing.

3.2 Facet types

The faceted taxonomy presented in an interface can contain several unrelated (or disjoint) sub-facets. For example, a book's price typically has nothing to do with its genre, and the construction and maintenance of the corresponding taxonomic structures are completely independent. We can refer to a facet, such as price and genre, as *a facet type*. An interface typically presents facets underneath a heading that indicates its type.[1] Within a facet type, facets are directly relatable and comparable. In other words, for our example, prices relate to prices and genres relate to genres; "$10-$20" can be a child of " <$100", but has no relationship with "horror". Figure 1 shows a facet type for patient demographics; Fig. 2 shows a facet type for medications. Demographics and medications are clearly disjoint: no taxonomic relations exist between these two types.

Definition 8 A facet type (a facet i and its related sub-facets) of a faceted taxonomy is a sub-category of **Tax**, the category of sets as objects and inclusion relations as morphisms. Let us call this sub-category $\mathbf{Facet}_i$ and let $Ob(\mathbf{Facet}_i) \subseteq Ob(\mathbf{Tax})$ with morphisms corresponding to $\subseteq$ relations for those objects. The relevant identity and composition definitions are also copied from **Tax**.

As seen in Fig. 3, the objects of $\mathbf{Facet}_i$ are sets which represent abstract collections of resources that have been classified to belong to that facet through faceted classification. With that being said, we do not need to distinguish between individual resources within these sets because they all categorically behave the same when manipulating objects: they either belong to a facet or they do not. Between the objects of $\mathbf{Facet}_i$, morphisms exist which dictate their taxonomic relationship. Figure 4 shows facet types in the context of a larger taxonomy; multiple facet categories exist and linkages between them may or may not exist.

As a concrete example, if $\mathbf{Facet}_{Med}$ is the medication facet-type displayed in Fig. 2, then {"central nervous system agent", "analgesic", "buprenorphine", …} $\in Ob(\mathbf{Facet}_{Med})$. The morphisms of $\mathbf{Facet}_{Med}$ dictate the relationships between objects. Suppose that object x is the set for "analgesic" and object y is the set for "central nervous system agent", then there exists a morphism $f: x \rightarrow y \in Hom_{\mathbf{Facet}_{Med}}(x, y)$, meaning "analgesic" *is-a* subset of "central nervous system agent".

The $\subseteq$ relation is powerful for specification: it allows for facets to be ordered by inclusion, which can model any structure where x is related to y; this is a pivotal component to most faceted implementations. We intentionally use the words specification and implementation; our category-theoretic model specifies the fundamental objects and relationships that our implementation can utilize. For example, we can implement a tree that only has *is-a* relationships since it is possible to order the facets by inclusion. We will show how this type of tree, which is heavily related to dynamic taxonomies (Sacco and Tzitzikas 2009), can be represented with category theory in Section 4. In a more complicated example, one could construct a visualization with a graph, where the current facet is recursively drawn connected to a subset of its subsumptive facets.

[1]One may want the interface to include the name of the type as a selectable facet. This meta-facet is mostly an organizational tool that aids in drawing the faceted taxonomy.

3.3 Focused selections

Given a facet, we need to describe how any selection within the facet can be modeled.

Definition 9 We can define a subcategory of **Facet**$_i$, called **Focus**$_i$, to represent a focused selection of objects from **Facet**$_i$ having Ob(**Focus**$_i$) ⊆ Ob(**Facet**$_i$) and the necessary corresponding morphisms, identity, and composition definitions for those objects.

We simply discard any undesirable objects (and their corresponding morphisms) to create a new category that represents a focused collection of facets. The identity and composition functions can be copied from **Facet**$_i$. Selecting objects in facet type is the simplest form of interacting with a faceted browser. As a concrete example, if **Facet**$_{Med}$ is the medication facet-type in Fig. 2, then one possibility is {“central nervous system agent”, “analgesic”} ∈ Ob(**Focus**$_{Med}$). There is no limit on the number of objects kept or discarded; it is possible to discard all objects or to keep all objects, although practical limitations may stem from the user's ability to interact with the interface.

3.4 Faceted taxonomies

Presentation of facets varies according to the interface's design, but facets are commonly presented as flat lists (through widgets such as radio buttons, check-boxes, etc) and hierarchies. Hierarchies, if restricted to be non-overlapping, are often represented as a tree where each facet is limited to having one parent. A non-hierarchical (or flat) facet is a special case of a hierarchical facet: it is simply a hierarchical taxonomy with one level. For example, if authors of books were a primitive facet, it could be represented as a hierarchy with author as the root and individual names as children. More complicated faceted interfaces may present taxonomies via visualizations, including graphs and network diagrams.

Definition 10 Let **FacetTax** be a category that represents a faceted taxonomy, whose objects are the disjoint union of **Facet**$_i$ categories. In other words, let $Ob(\mathbf{FacetTax}) = \coprod_{n=1}^{n} \mathbf{Facet}_i$ and n = |Ob(**FacetTax**)|. The morphisms of **FacetTax** are functors (mappings between categories) of the form $Hom_{\mathbf{FacetTax}}(\mathscr{C}, \mathscr{D}) = \{F : \mathscr{C} \rightarrow \mathscr{D}\}$.

This disjoint union is precisely how we can merge facets from Figs. 1 and 2 into a single faceted taxonomy, *patients*, as illustrated in Fig. 5. To save space, not all objects and relations are drawn. It is not necessary to formally prove that **FacetTax** is a true category because it is simply a subcategory of **Cat**, the category of categories. **FacetTax** acts as a highly structured version of **Tax**, where the objects are categories instead of objects; this additional structure allows us to model higher-level concepts such as faceted queries.

3.5 Universe and queries

The complexity of a faceted system naturally varies by the interface's design, but typically includes the ability to select (or focus) and de-select (or negate) facets within a facet type. The collective effort across all facets is then used to filter the faceted knowledge being presented.

Definition 11 A facet universe, U, is the n-ary product (Barr and Wells 1990) within the **FacetTax** category, defined as $\prod_{i=1}^{n} \mathbf{Facet}_i$, where $n = |Ob(\mathbf{FacetTax})|$. The n coordinates of U are projection functors $P_j : \prod \mathbf{Facet}_i \rightarrow \mathbf{Facet}_j$, where $j = 1,\ldots, n$ is the jth projection of the n-ary product.

Note that since $\mathbf{Focus}_i$ is a subcategory $\mathbf{Facet}_i$, there exists a restricted universe $U_{\subseteq} \subseteq U$ where every facet is potentially reduced to a focused subset. The act of querying the universe is essentially constructing this restricted universe $U_{\subseteq}$.

Definition 12 A faceted query, Q, is the modified n-ary product (Barr and Wells 1990) within the **FacetTax** category, defined as $\prod_{i=1}^{n} \mathbf{Focus}_i$, where $n = |Ob(\mathbf{FacetTax})|$. The n coordinates of Q are similarly defined as projection functors $P_j : \prod \mathbf{Focus}_i \rightarrow \mathbf{Focus}_j$.

A high-level overview of the interactions between $\mathbf{Facet}_i$, $\mathbf{Focus}_i$, and queries with the **FacetTax** category is illustrated in Fig. 6. To summarize, a faceted taxonomy is a category which contains **Facet** categories as objects; each **Facet** category contains objects and taxonomic relationships between the objects. Intuitively, focused selections are contained within their larger, unfocused facet categories. The universe of possible facets provided by the interface is the product across all **Facet** objects; a faceted query is simply a focused product chosen by the user from the universe of facets.

This implies that our model can represent states of interfaces that a user has interactively reached, which is a pivotal step in abstracting the interaction process within a faceted browsing system. Our model attempts to fill a void in human-computer interaction by providing abstractions that can consistently represent components of an interface for algebraic manipulation. Our model takes advantage of how intertwined a faceted browsing system is with its underlying faceted taxonomy and allows us to extend structural abstraction of a faceted taxonomy into the abstraction of faceted interfaces in general.

4 Reusable faceted taxonomies

In our model, categories are acting as generic faceted structures, but in practical interface development, more is sometimes needed to support the range of possible designs and interactions. The generic nature of the morphisms of **FacetTax** allow it to abstractly represent any faceted taxonomy structure since the morphisms are simply $\subseteq$ relations.

We demonstrated that n-ary products were one useful computation enabled by our model, but we can also demonstrate how our base structure can transform to support different faceted structures, such as lists, hierarchies, trees, graphs, and lattices. This transformation can be an active and engaging element of the interface: a selected element from a basic list could render a graph of deeper relationships. Using Fig. 5 as an example, suppose that a preview of high level facet types are given in the form of a list (*demographics, medications*, …) and interactively selecting one of these higher level items results in the rendering of its remaining taxonomy. For example, clicking on *demographics* could draw the descendants of that object from the taxonomy in the interface.

Intelligent previews are a possible solution to the problem of having too little screen space to fully display an interface's facets, which is a known issue in faceted browsing research (Wei et al. 2013). We can reuse the same facets, while manipulating their relations to fit other structures. These structures can be put into the same abstract framework to support interoperability between systems or between parts within a system. When designing a system where different structures interact with one another, the notation and representations become cumbersome. The burden of writing consistent abstractions with different structures is removed by using category theory.

4.1 Underlying graphs

In order to show how graphs relate and interact with our model, we must formally define the category of graphs, for which we follow (Spivak 2014).

Definition 13 Grph is the category with graphs as objects. A graph G is a sequence where $G := (V, A, src, tgt)$ with the following:

1. a set V of vertices of G
2. a set A of edges of G
3. a source function $src : A \rightarrow V$ that maps arrows to their source vertex
4. a target function $tgt : A \rightarrow V$ that maps arrows to their target vertex

An alternate way to represent a category is by noting it as a sequence of its constituent parts, e.g, a category $\mathscr{C} = (Ob(\mathscr{C}), Hom_{\mathscr{C}}, dom, cod, ids, comp)$, where $dom, cod : Hom_{\mathscr{C}} \rightarrow Ob(\mathscr{C})$ are domain and codomain functions, ids are identity functions, and $comp$ are compositions. This notation easily lets $\mathscr{C}$ be represented as a graph; in fact, some literature begins discussing categories as graphs first (Barr and Wells 1990); in this context, morphisms are typically called arrows and they map from a source (domain) to a target (codomain).

Definition 14 The graph underlying a category $\mathscr{C}$ is defined as a sequence $U(\mathscr{C}) = (Ob(\mathscr{C}), Hom_{\mathscr{C}}, dom, cod)$ (Spivak 2014).

Note that $U(\mathscr{C}) \in Ob(\mathbf{Grph})$ and there exists a functor between categories $\mathscr{F} : \mathscr{C} \rightarrow \mathscr{D}$ that can create a graph morphism $U(\mathscr{F}) : U(\mathscr{C}) \rightarrow U(\mathscr{D})$. This works at two levels for our model: for **FacetTax** and for each **Facet** category. Given that there exists a functor U: **Cat** $\rightarrow$ **Grph**, **FacetTax** can produce graphs of $\mathbf{Facet}_i$ categories for $i = (1,\ldots,|Ob(\mathbf{FacetTax})|)$.

Because every category has an underlying graph, each individual **Facet** can also be represented as a directed graph, where the objects are vertices and the morphisms are arrows from one vertex to another vertex. Although the taxonomy in Fig. 5 is likely best visualized as a simple hierarchy, other taxonomies might be more naturally suited to be represented as a graph. One possible use case is where graph algorithms might play a key role in the interface's design.

The key benefit to modeling our faceted taxonomy through category theory is that we can reuse the facets and re-frame their relationships (or morphisms) to fit our needs; if we need

to arrange the facets as graphs, we can do so. The sets of resources that the objects of **Facet** represent remain unchanged; resources are still classified with their facets as illustrated in Fig. 3. This embedded, faceted information can be reused through an alternate lens made possible by our model. In this case, the lens is a graph that was algebraically constructed, but other structures are also supported such as sets and lattices. Each structure has its own set of benefits; for example, graph representations allow the study of paths. In the next section, we will discuss paths and comment on using sets and lattices. Using category theory as a common language ensures interoperability within a model even when using different faceted structures.

4.2 Paths

When considering the graph underlying **FacetTax** or **Facet**$_i$, paths can play a role in modeling interactive components within an interface.

Definition 15 If $G := (V, A, src, tgt)$ is a graph, then a path of length n in G, is a head-to-tail sequence of arrows and is denoted $p \in \mathrm{Path}_G^{(n)}$, where $Path_G$ is the set of paths of any length in G (Spivak 2014).

In Fig. 7, we use paths as a simple example for modeling breadcrumbs, which have been identified as a key element of faceted search interfaces (Kules et al. 2009). In this case, a path within a focused subcategory represents a possible navigation route from the set of all navigation paths, $Path_{\mathbf{focus}}$. We will show in Section 5.3 that paths can be used in useful calculations within a category.

4.3 Sets

Rel is closely related to **Set**; both categories have sets as objects. In fact, it can be demonstrated that **Set** is a subcategory of **Rel** (Barr and Wells 1990) and we will utilize this notion to show that each **Facet**$_i$ is compatible with **Set**. We can construct a functor F: **Facet**$_i$ → **Set**, where F: $Ob(\mathbf{Facet}_i) \rightarrow Ob(\mathbf{Set})$ and for every $(x, y) \in Hom_{\mathbf{Facet}_i}(x, y)$ where x is related to y, we can construct a function $f(x) = y$ where an object is mapped to its ancestor in **Set**.

Similar to how we can construct graphs, we are also free to leverage **Set** categories and can construct basic sets. This is helpful when a flat list needs to be constructed from our taxonomy, such as in the case of the previews discussed in the first part of Section 4. In Section 5, we discuss several existing facet models that use set theory as a theoretical foundation and demonstrate how our categorical model can enable their reuse.

4.4 Other structures

Additionally, **Rel** is capable of representing lattices when ordering by inclusions (Barr and Wells 1990) and a similar result can be obtained with our **Facet** categories. Again, a functor is how we can compute such a mapping between categorical representations. Category theory is providing two types of computation: within a category with concepts, such as products, and across categories with concepts, such as functors.

4.5 Implementation

Even elegant abstractions can be rendered useless if they are not easily implementable. Category theory is strongly related to functional programming; one can even show that a functional programming language forms a category of types and operations (Barr and Wells 1990). Furthermore, the use of objects (and morphisms between objects) as our model's foundation means implementation could be in any programming language that supports an object-oriented paradigm. For instance, Scala (Odersky 2007) is a multi-paradigm language which supports both functional and object-oriented paradigms. In fact, the category of finite sets, often called **Fin** (Barr and Wells 1990; Spivak 2014), is easily implementable in Scala (Seeberger 2010); there are Scala libraries available that provide infrastructure for building categories (Hupel 2014a, b).

We contend that modeling has direct consequences for implementation; an inconsistent model yields an inconsistent system. If we cannot abstractly represent faceted browsing in an effective manner, it is difficult to extend and improve such a system and furthermore, the system cannot be adopted or easily modified by others. Theory must inform practice; we can use our abstractions to build stronger interfaces that support interoperability and reuse. Our model, together with category theory, can help inform how to build a proper application programming interface (API) for faceted browsing. In this vein, one can mathematically prove that something is possible before implementation. As illustrated in Fig. 8, an API for faceted browsing can provide methods for breaking down relationships from external file formats into primitive relationships (e.g., x *is-a* y) for abstract reasoning and logic. In the case of DELVE (Harris et al. 2014), faceted data from XML files are deconstructed into basic relationships; these relationships are then exported as a list in a JSON file that is compatible with a popular visualization library.

Our model does not intend to compete with existing and well-known standards such as the web ontology language (OWL) (McGuinness et al. 2004), but rather to easily marry it with other types and deconstruct its relationships into usable primitives for the development of faceted browsing interfaces. We aim to leverage faceted data in any format into a consistent framework for abstraction.

4.6 Computational complexity

We have focused on structural complexity between categories and morphisms, but computational complexity should also be addressed. It is only appropriate to consider computational complexity in the context of using abstract categories to write algorithms, which is one way our model can be applied and reapplied. An example is computing the transitive closure of **Fin**, which can be solved as the repeated accumulation of pairs of paths and can be demonstrated as having a complexity of $O(N^3 \log N)$ where N is the number of nodes (Rydeheard and Burstall 1988). Computational category theory (Rydeheard and Burstall 1988) connects functional programming with category theory to bridge the gap between theory and implementation. A by-product of computational category theory is that computational complexity can be studied.

5 Reusing existing facet models

We discussed several existing modeling efforts in our background section and we can now discuss how such models would change and manifest under our category-theoretic model.

5.1 Dynamic taxonomies

Dynamic taxonomies (Sacco and Tzitzikas 2009) use set theory as a theoretical foundation to help model facets. The model constructs taxonomies with *is-a* relationships by dynamically calculating the deep extension of a node:

$$\mathrm{deep}(C)=\{d|d \in \mathrm{shallow}(C') \wedge (C'=C \vee C' \mathrm{is\ \ a\ \ descendant\ \ of}\, C)\} \quad (1)$$

The shallow extension of C contains the direct descendants of C. Both shallow extension and *descendant-of* relationships are expressible as binary relations and, in particular, are expressible with the binary relations of **FacetTax**'s **Facet** categories. The shallow extension of an object $y \in Ob(\mathbf{Facet}_i)$ is the domain of the relations of $Hom_{\mathbf{Facet}_i}(x, y)$, e.g. all x that are subsets of y:

$$\mathrm{shallow}(y)=\bigcup_{\substack{x \in Ob(\mathbf{Facet}_i) \\ x \in y}} \mathrm{dom}(\mathrm{hom}_{\mathbf{Facet}_i}(x, y)) \quad (2)$$

Intuitively, we can think of the deep extension as the nested shallow extension, meaning we are free to construct $Hom_{\mathbf{Facet}_i}(x, dom(Hom_{\mathbf{Facet}_i}(y, z)))$ and so on. We can formalize this as a recursive union, so for any $y \in Ob(\mathbf{Facet}_i)$, we must look at all $x \in Ob(\mathbf{Facet}_i)$ where x is in the shallow extension of y.

In other words, to calculate the deep extension of y, we must recursively aggregate all x where x is a direct descendant of y:

$$\mathrm{deep}(y)=\bigcup_{\substack{x \in Ob(\mathbf{Facet}_i) \\ x \in \mathrm{shallow}(y)}} \mathrm{shallow}(y) \cup \mathrm{deep}(x) \quad (3)$$

The recursion stops when a leaf node is touched and the shallow extension of the object is the empty set. The leaf nodes only have an identity morphism where the domain and codomain is itself; this morphism is sometimes omitted when drawing taxonomic relations. We exclude this morphism by constraining the shallow extension to consider only those $x \in Ob(\mathbf{Facet}_i)$ where $x \neq y$, which in turn ensures that the recursion will end when calculating the deep extension. Ultimately for a leaf node y, there will be no $x \in Ob(\mathbf{Facet}_i)$ such that x is in the shallow extension of y. We are free to also include or exclude an object in the deep extension of itself, depending on what is most convenient for the faceted interface's design.

5.2 Category hierarchies

Category hierarchies are another example of facet models with set theory as a foundation (Li et al. 2010). In this model, category hierarchies are defined as connected and rooted directed acyclic graphs; the word category is unrelated to category theory in this context.

More specifically, a category hierarchy is defined as a sequence $H(r_H, C_H, E_H)$ representing a rooted and directed acyclic graph, where r_H is a designated root category, $C_H = \{c\}$ is the set of categories in the hierarchy, and $E_H = \{c \rightarrow c'\}$ is the set of category to subcategory relationships (Li et al. 2010). Facets are defined as subgraphs of the category hierarchy. We add a root category to our definition for underlying graphs in Section 4.1 and build a new sequence where $H(Ob(\textbf{FacetTax}), Hom_{\textbf{FacetTax}}, dom, cod, r)$ represents the category hierarchy. Facets are objects of **FacetTax** instead of direct subgraphs, but we can reuse the underlying graph notation to define a facet as a sequence $F_i(Ob(\textbf{Facet}_i), Hom_{\textbf{Facet}_i}, dom, cod, r_i)$ to create the graph underlying the $\textbf{Facet}_i$ category at some root r_i. At this point, we are free to leverage the deeper parts of the model using our facet graphs, such as measuring cost of navigational paths and facet similarity.

5.3 FaSet

There exist large differences between facet models even if they share the same theoretical foundation such as set theory. In FaSet (Bonino et al. 2009), a facet F is a set of items; in systems with multiple facets, they are disjoint: $F_a \cap F_b = \varnothing$. The items in the set represent labels for that facet. A focus L is a named subset of F: $L \subseteq F$, where the name is a nullable, variable-length list of indexes: $L\ \langle i, j, k, \ldots \rangle$. The classification of a resource r is the subset of F that is relevant, denoted as $r \perp F$. A sharp classification is a classification for some set of focus names P that can be expressed as a union of foci: $\exists P: r \perp F = \cup_{p \in P} L\ \langle p \rangle$. This model allows classifications to be written as easily implementable lists.

FaSet is purely a set-based model of faceted browsing. Differences between the models must be reconciled in order to inject the features of FaSet into **FacetTax**. The following differences must be noted:

1. The most primitive difference is that in FaSet a facet is a set of items (labels) while in **FacetTax**, a facet is a category. Many of the following differences are a direct result of **FacetTax** containing objects and relationships; mainly, it does not require external operators and special notations to reference relationships.
2. Because of FaSet's set-based foundation, its model must provide a set-based method of classifying resources with facets; it introduces custom notation to achieve this: $r \perp F$ for classifying a resource r with facet F. The objects of **FacetTax** are sets and represent abstract collections of resources that have been classified as belonging to that facet through faceted classification. In other words, **FacetTax** embeds the link to resources into its structure; the relationships between objects dictate the nature of their taxonomic relationships.

3. In FaSet, focusing is performed by creating lists of indexes on a facet. They introduce an indexing notation $L\ \langle i, j, k, \ldots \rangle$ for a focus of L where $L \subseteq F$. We abstract the concept of focusing and create a **Focus** category.

4. Because sets are flat structures, FaSet must and successfully does demonstrate that it is capable of handling facets with varying structure (flat, hierarchical, nested) and dimension (single or multiple). This is not necessary with our model: any object can hold any relationship with any other object. In a category-theoretic model, we do not need to name specific objects and local relationships are generic between each object: *x* is-a *y*, which can ultimately create flat, hierarchical, and nested structures.

The primitive aspects of FaSet are replaced by the structure naturally given by **FacetTax** and the related $\mathbf{Facet}_i$ categories. This helps avoid the need for creating custom notation. Once conceptual differences are reconciled, key elements and extensions of the model can be rephrased. For example, calculating focus similarity of two foci of a facet depends on finding the depth of a focus, which is the number of hierarchy levels that compose the name of the focus. There are many ways to calculate focus depth when using our model, but a natural way is to represent this depth as the largest path in the graph underlying $\mathbf{Focus}_i$, where paths are defined as in Section 4.1:

$$\mathrm{depth}(\mathbf{Focus}_i) = \max_{p \in \mathrm{Path}_{U(\mathbf{Focus}_i)}} (|p|) \tag{4}$$

5.4 Reuse and unification of efforts

Reusing existing models, or even parts of existing models, is a difficult endeavor due to the diversity of key concepts, definitions, and notations. Our goal in discussing how our model relates to existing work is to demonstrate how dynamic our category theory foundation can be and that it is able to drive interoperability and reuse. Resulting work can provide for the reuse and merging of existing models by uniting them in a common language. In this case, theory has direct consequences for specification and can help drive implementation.

6 Future work

We are applying our model to the next stage of our interface and framework, DELVE (Harris et al. 2014), in order to represent faceted structures that help create or control other faceted structures. Together with deeper elements of category theory, our model can help inform how to build a proper application programming interface (API) for faceted browsing. The role of computational category theory in developing this API is an interesting unknown.

We have demonstrated that it is possible to represent facets, faceted taxonomies, and faceted queries with category theory. Once the faceted query is performed, the interface must allow the user to successfully interact and engage with the faceted information being presented. We wish to continue our model to include this exploratory search phase of faceted browsing by leveraging deeper elements of category theory. For instance, what role do pull-back and

push-out operations play in modeling faceted browsing interfaces? Intuitively, they lend themselves to modeling interactivity, such as zooming.

For example, if an interface was designed to fulfill Shneiderman's information seeking mantra (Shneiderman 1996), how can tasks such as overview, filter, zoom, and details-on-demand be modeled? How can additional tasks from Shneiderman's task taxonomy (Shneiderman 1996), such as seeing relations, history, and extraction, be modeled?

Furthermore, if each of these tasks or interactions can be abstracted, does this model suggest anything about being able to measure the usability of the system? Usability is most commonly measured qualitatively, but we would like to explore the possibility of quantitatively measuring usability.

7 Conclusions

We have demonstrated that category theory can act as a theoretical foundation for faceted browsing that also encourages reuse and interoperability. We have established facets and faceted taxonomies as categories and have demonstrated how the computational elements of category theory, such as products and functors, extend the usefulness of our model.

The utility of faceted browsing systems is well-established in the digital libraries research community (Fagan 2013; Niu and Hemminger 2014), but current efforts could benefit from a more abstract framework that encourages reuse and interoperability. In this context, reuse and interoperability are at two levels: between systems and within a system. Our model works at both levels by leveraging category theory as a common language for representation and computation. Without this common language, it is difficult to abstractly model a system that utilizes multiple faceted structures (hierarchies, trees, graphs, lattices), even if there are shared notations and definitions between them.

We have demonstrated that category theory can be used to model faceted browsing and that it offers a consistent view of facets as objects and morphisms between objects. With our general framework for communicating mathematically about facets at a high level of abstraction, we can begin to construct interoperable interfaces and reuse existing efforts intelligently.

Acknowledgments

The project described was supported by the National Center for Research Resources and the National Center for Advancing Translational Sciences, National Institutes of Health, through Grant UL1TR000117. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH. This work would not be possible without support from Drs. Jerzy W. Jaromczyk, Todd R. Johnson, and Ramakanth Kavuluru.

References

Barr, M.; Wells, C. Category theory for computing science. New York: Prentice Hall; 1990.

Blass A. The interaction between category theory and set theory. Mathematical applications of category theory. 1984; 89:84–84.

Bonino D, Corno F, Farinetti L. Faset: A set theory model for faceted search. Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. 2009:474–481.

Chu HJ, Chow RC. An information model for managing domain knowledge via faceted taxonomies. Information reuse and integration (IRI), 2010 IEEE International Conference on. 2010:378–379.

Clarkson EC, Navathe SB, Foley JD. Generalized formal models for faceted user interfaces. Proceedings of the 9th ACM/IEEE Joint Conference on Digital libraries. 2009:125–134.

Coecke, B.; Paquette, ÉO. New Structures for Physics. Springer; 2011. Categories for the practising physicist; p. 173-286.

Dawson A, Brown D, Broughton V. The need for a faceted classification as the basis of all methods of information retrieval. Aslib proceedings, emerald group publishing limited. 2006; 58:49–72.

Fagan JC. Usability studies of faceted browsing: a literature review. Information Technology and Libraries. 2013; 29(2):58–66.

Fuentes-Lorenzo D, Morato J, Gómez JM. Knowledge management in biomedical libraries: a semantic web approach. Information Systems Frontiers. 2009; 11(4):471–480.

Giunchiglia, F.; Marchese, M.; Zaihrayeu, I. Journal on data semantics. Springer; 2007. Encoding classifications into lightweight ontologies; p. 57-81.

Harris DR, Kavuluru R, Yu S, Theakston R, Jaromczyk JW, Johnson TR. Delve: A document exploration and visualization engine. Proceedings of the Summit on Clinical Research Informatics. 2014:179.

Hearst MA. Clustering versus faceted categories for information exploration. Communications of the ACM. 2006a; 49(4):59–61.

Hearst MA. Design recommendations for hierarchical faceted search interfaces. SIGIR workshop on faceted search. 2006b:1–5.

Hupel, L. scalaz: functional programming for scala. 2014a. Retrieved January 25, 2015 from http://typelevel.org/projects/scalaz/

Hupel, L. scalaz category api documentation. 2014b. Retrieved January 25, 2015 from http://docs.typelevel.org/api/scalaz/stable/7.1.0-M3/doc/#scalaz.Category

Kules B, Capra R, Banta M, Sierra T. What do exploratory searchers look at in a faceted search interface? Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries. 2009:313–322.

Li C, Yan N, Roy SB, Lisham L, Das G. Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. Proceedings of the 19th international conference on World wide web. 2010:651–660.

McGuinness DL, Van Harmelen F, et al. Owl web ontology language overview. W3C recommendation. 2004; 10(10):2004.

Niu X, Hemminger B. Analyzing the interaction patterns in a faceted search interface. Journal of the Association for Information Science and Technology. 2014; doi: 10.1002/asi.23227

Odersky, M. The scala language specification, version 2.4 Technical report. Programming Methods Laboratory EPFL; Lausanne, Switzerland: 2007.

Phillips S, Wilson WH. Categorial compositionality: a category theory explanation for the systematicity of human cognition. PLoS Computational Biology. 2010; 6(7):e1000, 858.

Prieto-Díaz R. A faceted approach to building ontologies. Information reuse and integration, 2003 IRI 2003 IEEE International Conference on, IEEE. 2003:458–465.

Priss, U. Proceedings of the 8th International Conference on Conceptual Structures. Springer-Verlag; 2000. Faceted information representation; p. 84-94.

Rydeheard, DE.; Burstall, RM. Computational category theory. Englewood Cliffs: Prentice Hall; 1988.

Sacco, GM.; Tzitzikas, Y. Dynamic taxonomies and faceted search: theory, practice, and experience. Berlin: Springer; 2009.

Seeberger, H. Introduction to category theory in scala. 2010. Retrieved January 25, 2015 from http://hseeberger.github.io/blog/2010/11/25/introduction-to-category-theory-in-scala/

Shneiderman B. The eyes have it: a task by data type taxonomy for information visualizations. Visual languages, 1996 Proceedings, IEEE Symposium on. 1996:336–343.

Spivak, DI. Simplicial databases. 2009. arXiv:09042012

Spivak DI. Functorial data migration. Information and Computation. 2012; 217:31–51.

Spivak, DI. Category Theory for the Sciences. Cambridge: MIT Press; 2014.

Spivak DI, Kent RE. Ologs: a categorical framework for knowledge representation. PLoS One. 2012; 7(1):e24,274.

Spivak DI, Giesa T, Wood E, Buehler MJ. Category theoretic analysis of hierarchical protein materials and social networks. PLoS One. 2011; 6(9):e23,911.

Vickers P, Faith J, Rossiter N. Understanding visualization: A formal approach using category theory and semiotics. IEEE Transactions on Visualization and Computer Graphics. 2013; 19(6):1048–1061. [PubMed: 23559513]

Wei B, Liu J, Zheng Q, Zhang W, Fu X, Feng B. A survey of faceted search. Journal of Web engineering. 2013; 12(1-2):41–64.

Yee KP, Swearingen K, Li K, Hearst M. Faceted metadata for image search and browsing. Proceedings of the SIGCHI conference on human factors in computing systems. 2003:401–408.

Biography

Daniel R. Harris is an Application Systems Architect at the University of Kentucky's Center for Clinical and Translational Science and Institute for Pharmaceutical Outcomes and Policy. He received the B.S. degree in computer science and mathematics from the University of Kentucky, Lexington, KY, USA, in 2008, where he is currently a Ph.D. candidate in the Department of Computer Science. His research interests include biomedical informatics, interactive information systems, information visualization, and database systems.

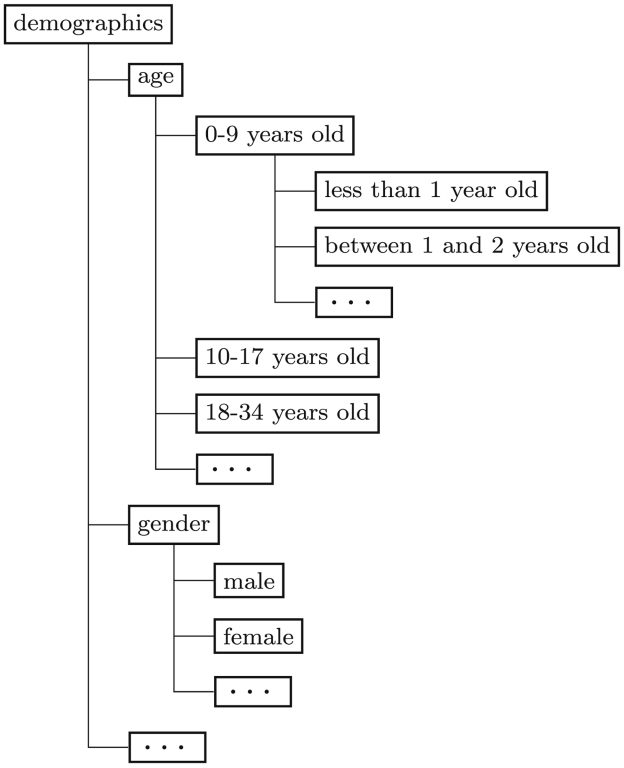


Fig. 1.

A small sample of facets available for patient medical records. For example, selecting female would query for all patients that are female

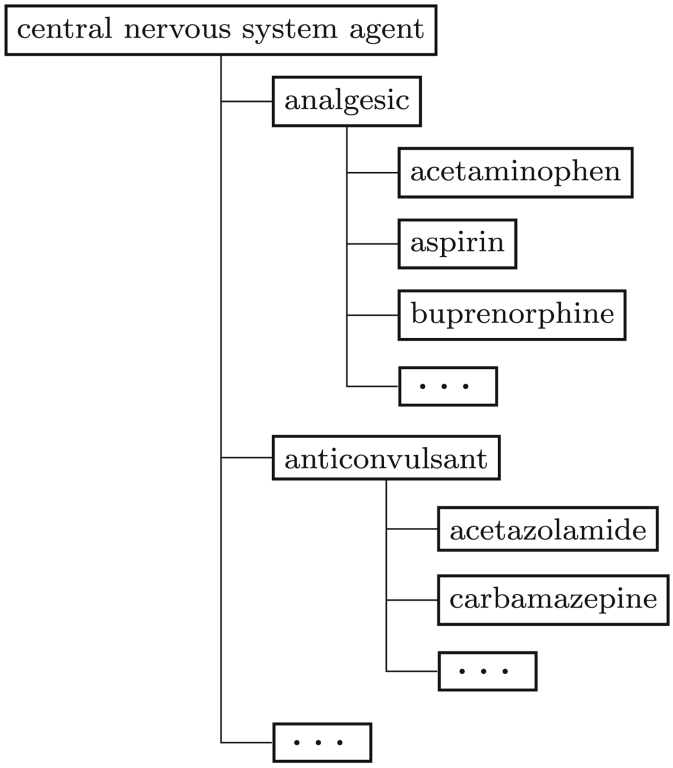


Fig. 2.
A small sample of facets available for drug administration records. For example, selecting analgesic would query for all records containing analgesics

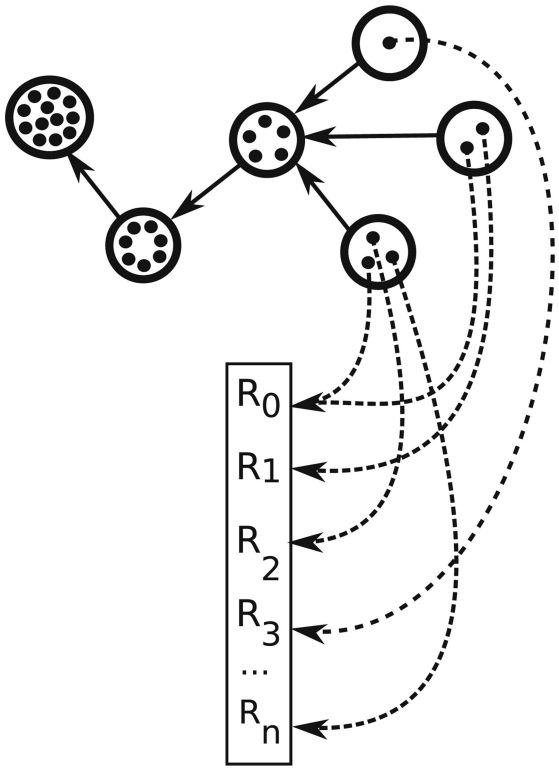


Fig. 3.

An illustrative example of a **Facet**$_i$ category shows a basic exemplative structure of the facets within this facet type. The objects are sets of pointers to resources classified as that particular facet type. For convenience, only pointers from leaf nodes have been drawn

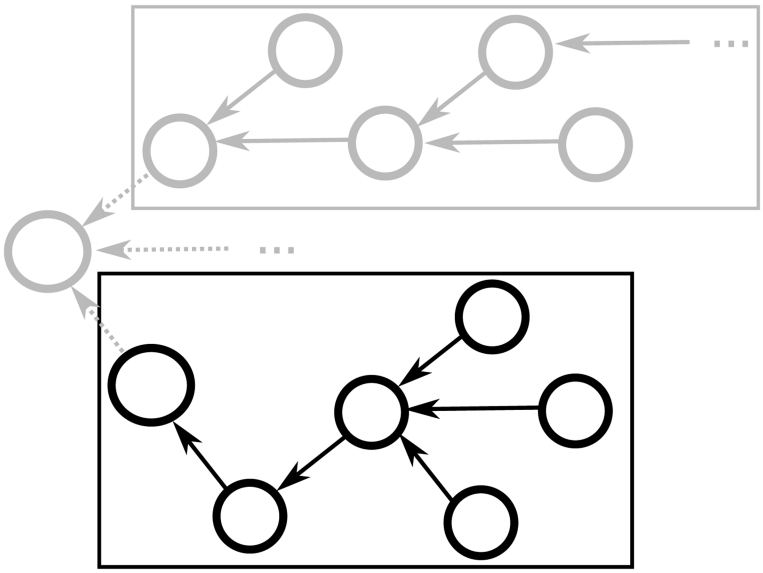


Fig. 4.

We draw the same exemplative facet type from Fig. 3 within the larger taxonomy. The facet types (illustrated as rectangular regions) act a grouping of objects that we can use as conceptual building blocks later in our model

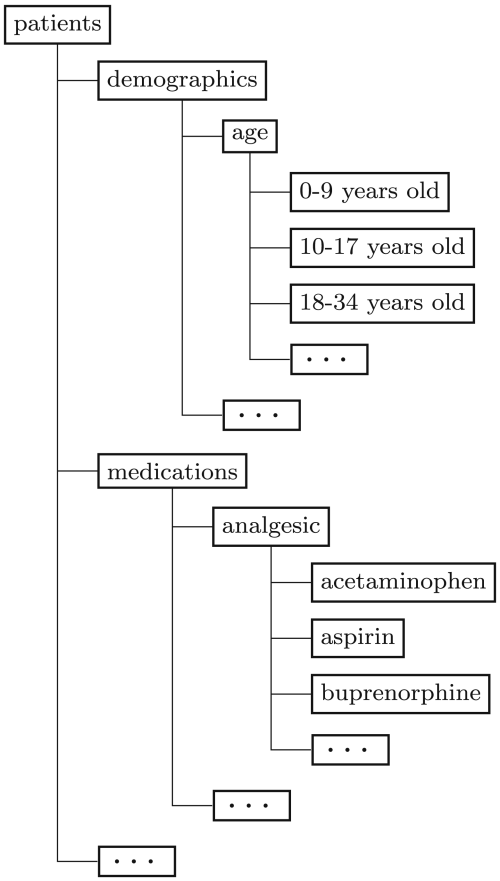


Fig. 5. A faceted taxonomy, called patients, containing facet types of demographics and medications

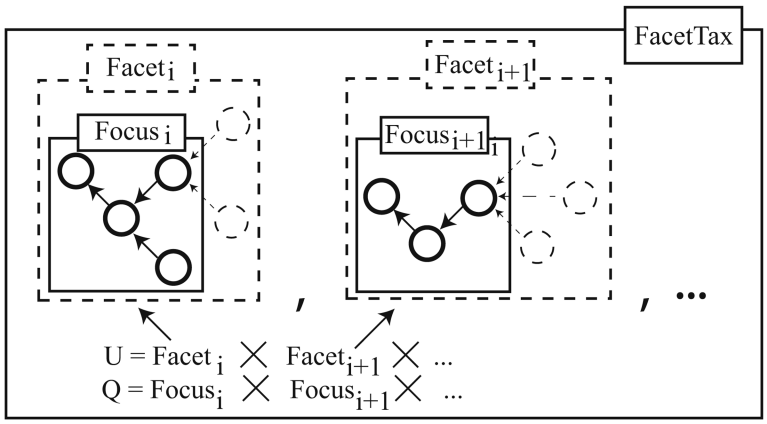


Fig. 6. A high-level overview of FacetTax, Facet$_i$, Focus$_i$, and queries with focused subcategories

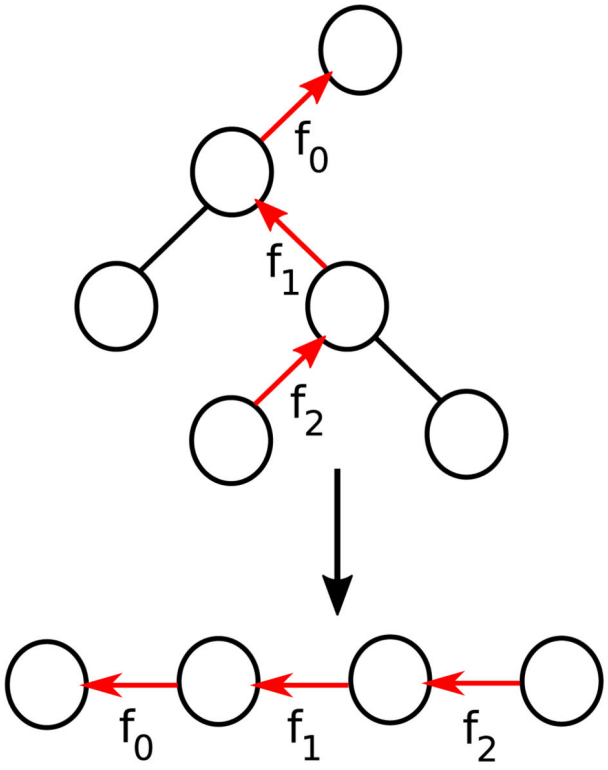


Fig. 7.

For graph-based faceted interfaces, paths are a simple way to abstractly model breadcrumbs. The above shows a breadcrumb f from a head-to-tail sequence of f_0, f_1, and f_2 arrows from a **Focus** category in **FacetTax**

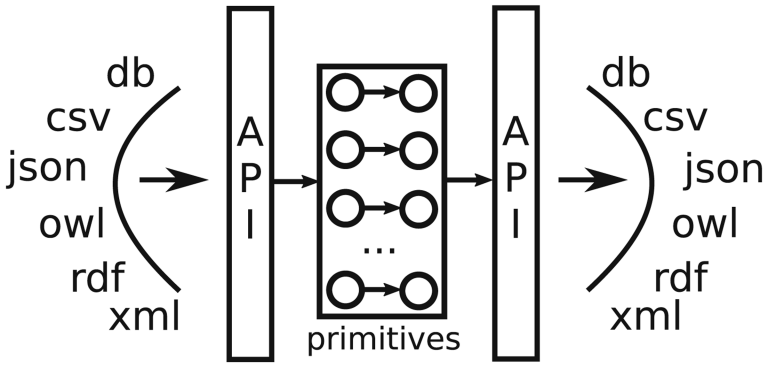


Fig. 8. An API can wrap our model of facets where incoming faceted data is broken into primitives for abstract reasoning and logic; we can then export in a convenient format for interface development