Using SPARQL to Test for Lattices: Application to Quality Assurance in Biomedical Ontologies

Guo-Qiang Zhang[1] and Olivier Bodenreider[2]

Case Western Reserve University, Cleveland, OH 44106, USA
National Library of Medicine, Bethesda, MD 20892, USA

Abstract. We present a scalable, SPARQL-based computational pipeline for testing the lattice-theoretic properties of partial orders represented as RDF triples. The use case for this work is quality assurance in biomedical ontologies, one desirable property of which is conformance to lattice structures. At the core of our pipeline is the algorithm called *NuMi*, for detecting the *Nu*mber of *Mi*nimal upper bounds of any pair of elements in a given finite partial order. Our technical contribution is the coding of *NuMi* completely in SPARQL. To show its scalability, we applied *NuMi* to the entirety of SNOMED CT, the largest clinical ontology (over 300,000 conepts). Our experimental results have been groundbreaking: for the first time, all non-lattice pairs in SNOMED CT have been identified exhaustively from 34 million candidate pairs using over 2.5 billion queries issued to Virtuoso. The percentage of non-lattice pairs ranges from 0 to 1.66 among the 19 SNOMED CT hierarchies. These non-lattice pairs represent target areas for focused curation by domain experts. RDF, SPARQL and related tooling provide an efficient platform for implementing lattice algorithms on large data structures.

1 Introduction

Lattices arise naturally from many disciplines. We speak of lattices because of their familiarity and their elegant structural properties. In the Semantic Web, lattices are intimately related to conceptual structures, since good ontologies often have a lattice structure [23]. The deeper philosophical and mathematical reason for lattice to be a desirable structural property for the taxonomy relation (e.g. IS-A) in ontologies can be elucidated using a theory called Formal Concept Analysis (FCA [3,11,15]). Starting from two very basic types, objects and attributes, with the assumption that intension and extension are fundamental adjoining facets of the notion of *concept*, one arrives at the mathematical structure of *(complete) lattices* automatically using FCA. The upshot of this is that if we encounter a non-lattice fragment in a taxonomic hierarchy, then somewhere upstream the notion of intension and extension has not been rigorously enforced, revealing gaps in conceptual modeling which can be subtle to detect otherwise.

Though desirable, the lattice property of ontologies is not always found in most biomedical ontologies. For example, SNOMED CT, the largest clinical ontology, is only a lattice "for the most part" (assuming superficial top and bottom elements). As can be seen from Fig. 1, the double-circled concepts "Tissue specimen from breast" and "Tissue specimen from heart" legitimately share the two features of being a kind

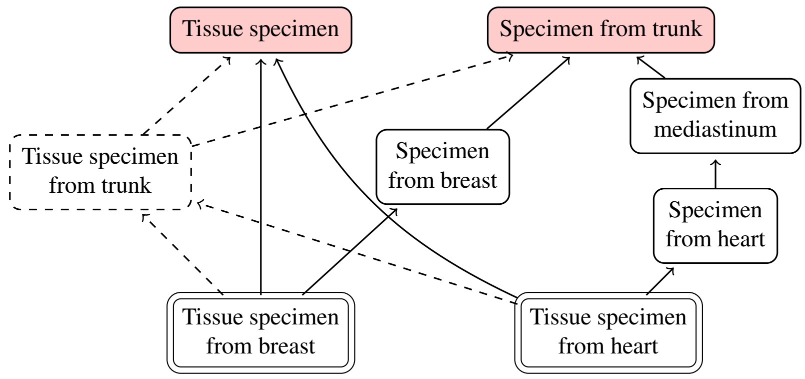

Tissue specimen
Specimen from trunk
Specimen from mediastinum
Tissue specimen from trunk
Specimen from breast
Specimen from heart
Tissue specimen from breast
Tissue specimen from heart

Fig. 1. Non-lattice fragment from the Specimen hierarchy in SNOMED CT. Dashed lines represent a possible remedy by adding the concept "Tissue specimen from trunk".

of tissue specimen and a kind of specimen from trunk. The current representation in SNOMED CT involves two minimal upper bounds shared by the two double-circled concepts, "Tissue specimen" and "Specimen from trunk", which is the reason why this fragment is not part of a lattice. The corresponding lattice-conforming representation would require the creation of the concept "Tissue specimen from trunk" (the dashed component in Fig. 1), which would be the single minimal (least) upper bound of the two double-circled concepts. Incidentally, the concept pair "Tissue specimen from breast" and "Tissue specimen from heart" is one of the 28,464 non-lattice pairs found by this work in the Specimen hierarchy (see Table 1).

Since the majority of biomedical ontologies are manually curated, automatic testing for lattices plays an important role in auditing and quality assurance of these ontological systems. The absence of a lattice structure can be indicative of issues including missing concepts, misaligned concepts and inconsistent use of pre-coordination [7,19]. However, it is also possible that the clinical utility of some concepts was deemed insufficient to warrant their creation.

We present a scalable, SPARQL-based computational pipeline for testing the lattice-theoretic properties of partial orders represented as RDF triples. The basic idea is to simply follow the definition of a lattice and check that each pair of elements has a least upper bound. If this is true, then each pair of elements also has a greatest lower bound, by pure mathematical reasoning (Proposition 1). Thus to check for lattices, we only need to check either the existence of least upper bounds, or the existence of greatest lower bounds, for the whole partial order. For convenience, we will focus on least upper bounds in this paper, but all results translate directly to the situation of (testing for existence of) greatest lower bounds. In practice, for finite partial orders, including the taxonomic backbones of ontological systems, the existence of a least upper bound for a pair is equivalent to the uniqueness of minimal upper bounds of the given pair.

With brute force, checking for the number of minimal upper bounds can be computationally daunting for large ontological systems. For example, the July 31, 2009 version of SNOMED CT comprises 307,754 (N) active concepts, with maximal depth of 30. If for each of the 47,356,108,381 [(N*(N-1))/2] pairs we (1) find all their upper bounds and (2) detect the minimal ones among the upper bounds, the computation quickly become intractable. Assuming each pair takes 10 ms (a reasonable estimate), processing the whole SNOMED CT would take about 15 years.

The main contribution of this paper is a demonstration of the suitability of Semantic Web technologies, namely RDF and SPARQL, for quality assurance in large biomedical ontologies by testing their lattice-theoretic properties. We have discussed the clinical significance of our work in [19]. Here, we focus on general technical aspects. We have developed an algorithm called *NuMi*, for detecting the *Nu*mber of *Mi*nimal upper bounds of any pair of elements in a given partial order, which we have implemented completely in SPARQL. We also propose further optimization by applying a reverse version of the algorithm. The experimental results reported here are groundbreaking: for the first time, we have been able to exhaustively check the entirety of SNOMED CT for its lattice-theoretic properties within a time frame of 2 months sequential computation. The percentage of non-lattice pairs ranges from 0 to 1.66 (Table 2) among the 19 SNOMED CT hierarchies among over 34 million candidate pairs. These pairs represent potential target areas for focused curation by domain experts. The reason that testing the lattice-property allows us to efficiently and systematically identify targets for curation stems from the rationale broadly captured in FCA [3,11,15] and indirectly through the work of Jiang and Chute [7] as well.

2 Background

2.1 SNOMED CT

SNOMED CT is a comprehensive concept system for healthcare, distributed and maintained by the International Health Terminology Standard Development Organization (IHTSDO) [6]. SNOMED CT provides broad coverage of clinical medicine, including findings, diseases, and procedures, and is used in electronic medical records [1].

The development of SNOMED CT is supported by an infrastructure based on description logics. From a structural perspective, SNOMED CT can be seen as a series of large directed acyclic graphs, one for each of its 19 "hierarchies": Procedure, Physical force, Event, Staging and scales, Substance, Environment or geographical location, Situation with explicit context, Body structure, Observable entity, Pharmaceutical / biologic product, Physical object, Qualifier value, Special concept, Specimen, Social context, Clinical finding, Organism, Linkage concept, and Record artifact. No concept is shared across hierarchies except for the root. Each concept comes with a SNOMED CT ID, which is an integer such as those given in the first column of Table 1. SNOMED CT concepts are linked by hierarchical relations, within each hierarchy (e.g., "Tissue specimen from heart" IS-A "Tissue specimen"). Associative relations (across hierarchies) form the basis of the logical definitions (e.g., "Nephrectomy" procedure site "Kidney"), but are not used in this work. The version of SNOMED CT used in this study is dated July 31, 2009 and comprises 307,754 active concepts.

The motivation for this work comes in part from the application of Formal Concept Analysis (FCA) to a limited subset of SNOMED CT by Jiang and Chute [7]. These authors used FCA as an auditing tool by constructing local contexts from normal form presentations in SNOMED CT (i.e., logical definitions in description logic) and compared with the resulting lattices for anonymous (unlabeled) nodes. They showed that given a small SNOMED CT fragment, this method can automatically identify a candidate pool of missing concepts for further examination by domain experts. However, constructing lattices from contexts is so computationally expensive that it is hardly scalable [10,15]. FCA-based analysis is therefore not applicable to the entirety of large ontologies such as SNOMED CT. Moreover, for ontological systems without rich logical definitions, the need for background contexts would render the FCA-based approach inapplicable.

2.2 Lattices, Complete Lattices and Quasi-primes

We first review basic definitions in lattice theory. Our main references are [5,20].

A partially ordered set (poset) is a set L with a reflexive, transitive relation $\leq\, \subseteq L \times L$. If $(L, \leq)$ is a poset, then its dual is the poset $(L, \geq)$. We denote posets by their carrier set as long as the partial order is clear from the context. An element u is called a *upper bound* of a subset $X \subseteq L$, if for each $x \in X$ we have $x \leq u$. For convenience, we write $\mathsf{ub}(X)$ for the set of upper bounds of X. An element m is called an *minimal upper bound* of a subset $X \subseteq L$, if m is an upper bound of X, and for any $n \leq m$ such that $x \leq n$ for each $x \in X$, we have $m = n$. We write $\mathsf{mub}(X)$ for the set of minimal upper bounds of X. When $\mathsf{mub}(X)$ is a singleton, the unique minimal upper bound is called the least upper bound, or join, of X. The notion of lower bound, maximal lower bound, greatest lower bound (meet), is defined dually. Specifically, $\mathsf{mlb}(X)$ represents the set of maximal lower bounds of X.

A poset L is a *lattice* if every two elements of L have a join and a meet. These meets and joins of binary sets will be written in infix notation: $\bigvee\{x, y\} = x \vee y$ and $\bigwedge\{x, y\} = x \wedge y$. A poset L is a *complete lattice* if every subset $S \subseteq L$ has a least upper bound $\bigvee S$ (join) and a greatest lower bound (meet) $\bigwedge S$.

Fig. 2 contains the diagram of a small poset which is not a lattice. Note that in this poset, we have $\mathsf{ub}\{a, b\} = \{x, y, \top\}$, and $\mathsf{mub}\{a, b\} = \{x, y\}$, i.e. x and y are minimal upper bounds of a and b. Hence the pair $\{a, b\}$ does not have a least upper bound and this diagram does not represent a lattice. When the size of $\mathsf{mub}\{a, b\}$ is greater than 1, we call a, b a non-lattice pair (e.g., a, b in Fig. 2).

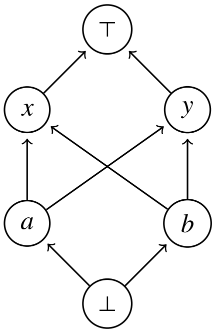


Fig. 2. A poset which is not a lattice

In connection with ontology, one can think of concepts as elements of a poset, and the ordering relation as the subsumption relation [8]. If x, y are concepts, we write $x \leq y$ to mean x IS-A y, or y subsumes x. The join $x \vee y$ of two concepts x, y is the lowest common ancestor of x and y, and the meet $x \wedge y$ is their greatest common descendant.

Every finite lattice is a complete lattice. Therefore, every finite lattice has a top (largest) element, denoted as $\top$, and a bottom (least) element, denoted as $\perp$. One can think of a tree as a lattice by adjoining a superficial bottom. In this sense, lattices are more general than trees, but posets are more general than lattices. Multiple inheritance is not permitted in trees: each node in a tree can have at most one parent (the node immediately above it). Lattices permit multiple inheritance but insist on the existence of (unique) join and meet for any pair of nodes. Since adding top and bottom elements can be globally achieved either conceptually or materialized, we assume that posets come with top $\top$ and bottom $\bot$ in the remainder of the paper. By convention, we have $\bigvee \emptyset = \bot$ and $\bigwedge \emptyset = \top$.

If every pair of elements in a finite poset has a least upper bound $\vee$, then the greatest lower bound of a pair a, b can be obtained as $a \wedge b := \bigvee\{x \mid x \leq a \text{ and } x \leq b\}$.

This leads to the well-known property of "half-implies-whole" in the next proposition, which forms the basis of our computational strategy later.

Proposition 1. *Let $(L, \leq)$ be a finite poset. If any pair of elements in L has a least upper bound, then L is a lattice.*

In a poset $(L, \leq)$, an element x is *covered* by y if $x \leq y$, $x \neq y$, and for any z such that $x \leq z \leq y$, we have either $z = x$ or $z = y$. To reduce storage, it is often the case that only the coverage relation is stored explicitly for ontological systems such as SNOMED CT.

The notion of quasi-prime [21] will be useful later on for selecting candidate pairs for our SPARQL query. An element q in a poset $(L, \leq)$ is called a *quasi-prime* if for any subset $X \subseteq L$, $q \in \mathsf{mlb}(X)$ implies $q \in X$ (when $\bigwedge$ exists for X, we have $\mathsf{mlb}(X) = \{\bigwedge X\}$). Since we will be concerned with finite posets with top and bottom elements, the combinatorial interpretation of quasi-primes for finite posets will be useful. Note that the top element of a poset cannot be a quasi-prime, since $\top = \bigwedge \emptyset$ but $\top \notin \emptyset$. The notion of coquasi-prime can be defined similarly.

Proposition 2. *An element q in a poset $(L, \leq)$ is a quasi-prime if and only if there exists $q^* > q$, such that for any $x > q$, we have $x \geq q^*$.*

Proposition 2 implies that for each quasi-prime q, there is a unique element (q^*) covering q, or q has a single parent node. Moreover, for any $a \in L$ such that neither $a \leq q$ nor $a \geq q$ holds, if x is an upper bound of $\{a, q\}$, then x must already be an upper bound of $\{a, q^*\}$. As a consequence, we have $\mathsf{mub}\{a, q\} = \mathsf{mub}\{a, q^*\}$.

Proof. We provide a general proof without assuming L to be finite.

(Only If). For a quasi-prime $q \in L$, consider the set $U_q := \{x \in L \mid x > q\}$. Since the top element is not a quasi-prime, U_q is not empty. If no lower bound of U_q is strictly above q, then $q \in \mathsf{mlb}(U_q)$ but $q \notin U_q$, which is impossible for a quasi-prime q. Therefore, there exists a lower bound q' of U_q such that $q < q'$. Since $q' \in U_q$ by definition of U_q, q' must be the least element in U_q. We take the required q^* to be q'.

(If). Suppose there exists $q^* > q$, such that for any $x > q$, we have $x \geq q^*$. Suppose $q \in \mathsf{mlb}(X)$ for some $X \subseteq L$. X cannot be empty since otherwise $q = \top$, and there cannot be an element q^* strictly above top. If $q \in X$, then there is nothing to prove. Otherwise, for any $x \in X$ we have $q < x$, and so $q^* \leq x$ by assumption. This means q^* is a lower bound of X strictly dominating q, which contradicts the assumption that q is a maximal lower bound ($q \in \mathsf{mlb}(X)$). Therefore, we must have $q \in X$.

As a consequence of Proposition 2, quasi-primes are not needed in detecting non-lattice properties, since if there is any violation of the uniqueness of minimal upper bounds involving a quasi-prime q, there must already be such a violation involving q^*. In Fig. 1, the concepts "Specimen from breast", "Specimen from heart" and "Specimen from mediastinum" are quasi-primes with visually identifiable q^*'s in the same diagram. This leads to the following definition.

Definition 1. *A pair a, b in a poset $(L, \leq)$ is called a probe pair (probe, for short) if*

1. *neither a nor b is a quasi-prime;*
2. *a and b are not comparable, i.e., neither $a \leq b$ nor $b \leq a$ holds.*

In Section 4, our quality assurance use case will significantly reduce the candidate pairs to be tested by running *NuMi* on probes only.

2.3 RDF and SPARQL

The Resource Description Framework (RDF) is a directed, labeled graph data format for representing information in the Web [12]. Based on (subject, predicate, object) triples, RDF is well suited for the representation of graphs in general, including posets and lattices. Because of its origins in the Semantic Web, RDF uses Unified Resource Identifiers (URIs) as names for the nodes and the links in the graph.

SPARQL is a query language for RDF graphs [13]. SPARQL queries are expressed as constraints on graphs, and return RDF graphs or sets as results. For example, SPARQL can be used for retrieving the set of common ancestors of two nodes in a graph, i.e., to compute the upper bounds for a pair of nodes from the graph. In practice, as shown in Fig. 3, the list of direct ancestors common to two nodes a and b can be easily obtained by querying the nodes of which both classes are a subclass. The same variable (?upper) is used in the constraints imposed to the graph for the two nodes a and b. This pattern forms the basis for testing the lattice-theoretic properties in the queries we developed for this work.

```
SELECT ?upper
WHERE {
       :a rdfs:subClassOf ?upper.
       :b rdfs:subClassOf ?upper.}
```

Fig. 3. SPARQL query for common ancestors of a and b

Although developed in a description logic environment, SNOMED CT is distributed as a set of relational tables and there is no version of SNOMED CT available in RDF. We transformed SNOMED CT into RDF using a simple script. A base URI was added to SNOMED CT identifiers in order to create URIs for concepts and predicates. The fully-specified name was used as the label for the concepts. All relations among concepts were transformed into triples, using `rdfs:subClassOf` for representing the native IS-A relationship.

Additionally, we precomputed the transitive closure of `rdfs:subClassOf`, because a transitively-closed graph was assumed in some aspects of our algorithm to both improve speed and avoid computing transitive closure on the fly, which is not supported in most existing SPARQL environments.

3 Methods

In this section we first present a simple algorithm in the conventional style for computing the mub set (see Section 2.2) mub$\{a, b\}$ for a given pair a, b within a finite poset $(L, \leq)$. We then introduce a SPARQL implementation of the algorithm with the input poset viewed as a graph, represented as RDF triples. Finally, we revisit the original algorithm and propose an optimization.

3.1 Algorithm for Identifying Minimal Upper Bounds

The following algorithm finds the number of minimal upper bounds of a, b by keeping track of "counts" of the number of times an element in L occurs as an upper bound of a, b. After *count* is initialized (lines 1-3), every upper bound of a, b gets *count* incremented by 1 (lines 4-8). So by line 8, the counts for each upper bound of a, b is precisely 1, and we have ub$\{a, b\} = \{x \mid count(x) = 1\}$, at this point. The third iteration (lines 9-13) increases counts for those members in ub$\{a, b\}$ that are *not* minimal within ub$\{a, b\}$ (i.e., those elements that are strictly above some other element in ub$\{a, b\}$). At the end of this iteration (line 13), we have mub$\{a, b\} = \{x \mid count(x) = 1\}$.

```
   Data: Elements $a, b$ in a finite poset $(L, \leq)$
   Output: The size of mub$\{a, b\}$
1  for each $x \in L$ do
2      $count(x) := 0$
3  end
4  for each $u \in L$ do
5      if $u$ is an upper bound of $\{a, b\}$ then
6          $count(u) := count(u) + 1$
7      end
8  end
9  for each $v \in$ ub$\{a, b\}$ do
10     if $v > u$ for some upper bound $u$ of $\{a, b\}$ then
11         $count(v) := count(v) + 1$
12     end
13 end
14 Count the number of $x \in L$ with $count(x) = 1$ and output this number
```

Algorithm 1. *NuMi* in the conventional procedural style, for finding the number of minimal upper bounds.

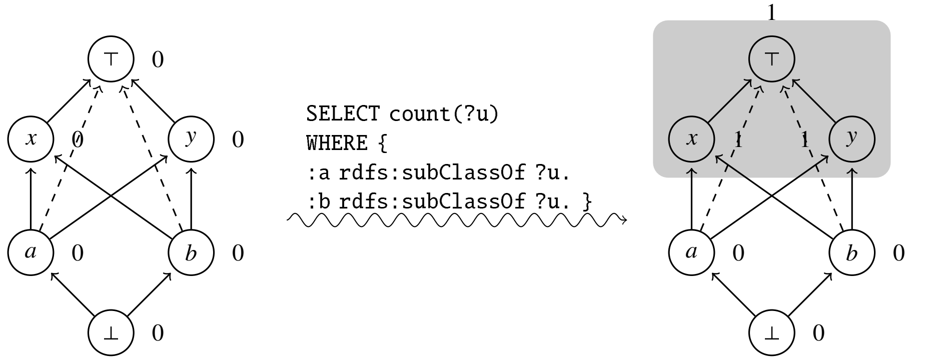
The correctness of *NuMi* is self-evident. To check for lattices, one runs *NuMi* on every pair a, b from an input poset L for potential violations of the lattice property. If all pairs have exactly one minimal upper bound, then L is a lattice; otherwise, pairs with more than one minimal upper bound will be identified as non-lattice pairs.

3.2 Implementing *NuMi* in SPARQL

We describe a method to implement *NuMi* completely in SPARQL. This allows us to take advantage of the highly optimized storage and access environment of RDF stores in existing systems such as Virtuoso. To implement Algorithm 1, we construct a two-part SPARQL query, corresponding to two iterations in Algorithm 1: (1) to compute $\mathsf{ub}\{a, b\}$ (lines 4-8), and (2) to compute $\mathsf{mub}\{a, b\}$ (lines 9-13).

Although aggregation operators such as `count` are not part of the SPARQL 1.0 specification, they are already available in many SPARQL environments, including Virtuoso. In contrast, ancestor tracing and computing transitive closure are not supported in most of existing SPARQL environments. Both for this reason and for saving computational time, we decided to *precompute the transitive closure* of `rdfs:subClassOf` in RDF store for the input graph for L. In other words, the RDF store, for which the SPARQL queries are made, is assumed to be transitively closed.

The first part of the query finds all upper bounds $?u$, and tracks the result by having each upper bound to receive 1 as count. This is straightforward using the query fragment indicated in the middle of Fig. 4.



```
SELECT count(?u)
WHERE {
:a rdfs:subClassOf ?u.
:b rdfs:subClassOf ?u. }
```

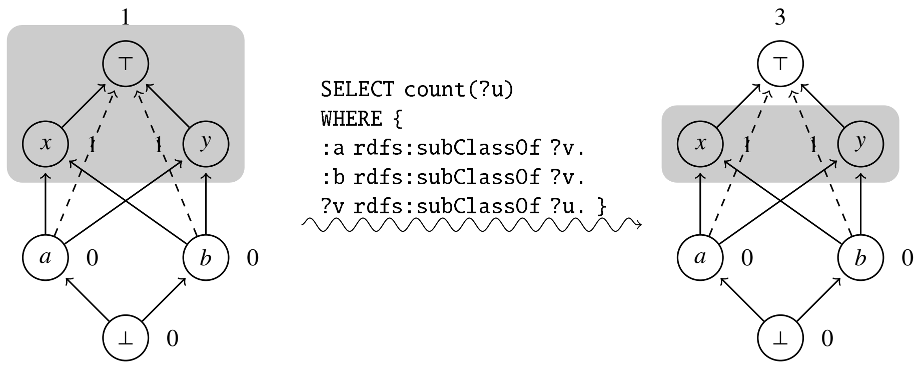
Fig. 4. SPARQL query for finding $\mathsf{ub}\{a, b\}$ indicated in the shaded area. Dashed edges represent those due to the effect of transitive closure.

The second part of the query finds elements $?u$ that are strictly above some elements in $\mathsf{ub}\{a, b\}$. This can be achieved by the query fragment indicated in the middle of Fig. 5.

Joining the two query fragments with the `union` operator, we obtain a complete SPARQL query for finding the minimal upper bounds $\mathsf{mub}\{a, b\}$. Fig. 6 displays a complete working SPARQL query.

3.3 Optimization: Reverse SPARQL Query

In earlier sections, we described optimization strategy by skipping pairs of concepts that (1) do not belong to the same SNOMED CT sub-hierarchy; (2) are in hierarchical relationship to one another (i.e., comparable); (3) involve a quasi-prime. This subsection describes a strategy that checks for the existence of greatest lower bounds, rather than least upper bounds, achieved by running SPARQL queries “in reverse.”



```
SELECT count(?u)
WHERE {
:a rdfs:subClassOf ?v.
:b rdfs:subClassOf ?v.
?v rdfs:subClassOf ?u. }
```

Fig. 5. SPARQL query for finding mub$\{a, b\}$ indicated in the shaded area on the right side

```
SELECT  ?sb  count(?sb) as ?sb_links
FROM <http://newton.case.edu/TEST>
WHERE {
  {
      <http://mor.nlm.nih.gov/SNOMEDCT#256889002> rdfs:subClassOf ?sb.
      <http://mor.nlm.nih.gov/SNOMEDCT#258462005> rdfs:subClassOf ?sb.
  }
  union
  {
      <http://mor.nlm.nih.gov/SNOMEDCT#256889002> rdfs:subClassOf ?sa.
      <http://mor.nlm.nih.gov/SNOMEDCT#258462005> rdfs:subClassOf ?sa.
      ?sa rdfs:subClassOf ?sb.
  }          }
ORDER BY ASC (?sb_links)
```

Fig. 6. Example SPARQL query for SNOMED CT concepts 256889002 and 258462005

On average, concepts in ontological hierarchies tend to have fewer upper level concepts representing more general and abstract entities, and more lower level concepts.

Since lattices can be viewed either top-down or bottom-up, only one direction (i.e., among least upper bounds or greatest lower bounds) needs to be tested (see Proposition 1). For distinct elements u, v, x, y in a finite poset, if $x, y \in \mathsf{mub}(\{u, v\})$ then $u, v \in \mathsf{mlb}(\{x, y\})$ in most cases, as illustrated in the top part of Fig. 7. Since there are generally more concepts lower in a hierarchy (as in SNOMED CT's taxonomic backbone), this motivates us to test for maximal bounds (mlb) in the original order $(L, \leq)$, or equivalently, minimal upper bounds (mub) in the reverse order $(L, \geq)$, to reduce the number of non-lattice pairs. The following proposition ensues that even though the lattice and non-lattice status of a given pair is usually different when the order is reversed, a non-lattice pair in the original order is guaranteed to appear in a "fragment" generated by some non-lattice pair in the reverse.

Proposition 3. *Suppose* $x, y \in \mathsf{mub}\{a, b\}$, *where* x, y, a, b *are distinct elements in a finite poset* $(L, \leq)$. *Then there exists* $u, v \in L$, *such that* $a \leq u$, $b \leq v$, *and*

$$x, y \in \mathsf{mub}\{u, v\} \text{ and } u, v \in \mathsf{mlb}\{x, y\}.$$

The reverse query in SPARQL for the lower bounds (i.e., common descendants) of nodes x and y is straightforward. It is obtained by switching the position of variables and input nodes in the query (at the bottom part of Fig. 7).

4 Quality Assurance Pipeline for SNOMED CT

We applied the SPARQL implementation of *NuMi* to auditing SNOMED CT by systematically identifying all non-lattice fragments. The following phases were involved in this quality assurance study: (1) acquiring SNOMED CT data; (2) selecting probes; (3) testing probes; (4) summarizing and analyzing results.

$$\begin{tikzcd}
x & y \\
u \arrow[u] \arrow[ur] & v \arrow[u] \arrow[ul]
\end{tikzcd}$$

```
SELECT ?lower
WHERE {
      ?lower rdfs:subClassOf :x.
      ?lower rdfs:subClassOf :y.}
```

Fig. 7. Template for reverse SPAQRL query to obtain lower bounds

4.1 Acquiring SNOMED CT Data

From the 307,754 active concepts in SNOMED CT, we created RDF triples for representing the IS-A relations. We created URIs for all SNOMED CT concepts and used the `rdfs:subClassOf` predicate to represent the IS-A relationship. Then we computed the transitive closure of the IS-A relation and created a distinct set of triples for it. The graph of hierarchical relations contains a total of 439,733 `rdfs:subClassOf` triples, while the transitively-closed graph contains 1,191,796 triples. The two sets of triples were loaded into two separate graphs using the open source Virtuoso triple store [16].

4.2 Selecting Eligible Probes

Not every pair of SNOMED CT concepts needs to be tested for its lattice properties. For efficiency reasons, we start by selecting eligible probes for further testing. Since the 19 hierarchies in SNOMED CT do not share any concepts (except the root), only pairs within the same hierarchy require testing. In addition, only probe pairs (see Definition 1) need to be tested for the lattice property, since pairs in which each concept does not have at least two parent concepts will always have their evidence as part of a non-lattice structure exhibited by pairs without involving a quasi-prime (see Subsection 2.2). Moreover, any pair in which one concept is an ancestor of the other does not need to be tested because the ancestor concept in such pairs is the unique common ancestor of the two, with reflexivity assumed. In practice, we constructed SPARQL queries to implement the eligibility criteria described above and ran them in order to establish the list of probes to be tested for lattice properties. Because reverse queries operate on the graph of descendants instead of that of ancestors, eligibility criteria for probes are slightly different. More specifically, probe concepts are required to have at least two child concepts (instead of at least two parent concepts).

4.3 Testing Probes Using SPARQL

The algorithms presented earlier for selecting the probes to be tested, and testing them, were implemented without any *ad hoc* programming. Generic queries were created for each algorithm and subpart thereof and loaded as stored procedures in Virtuoso. Stored procedures are used to compensate for the fact that SPARQL queries could not be "prepared" (i.e., the query plan cannot be cached by the ODBC driver in our environment), which is a serious limitation, as about half of the query execution time is generally devoted to building the query plan.

We used a simple script to compute the cartesian product of all pairs of concepts within a given hierarchy of SNOMED CT. Each pair was evaluated as a potential probe by querying a stored procedure instantiated with the pair. If qualified as a probe, the pair was then tested using *NuMi* by querying a second stored procedure. The results were stored in text files for further processing. The open source Virtuoso RDF store version 06.00.3123 was used for this experiment, running on a Dell 2950 server (Dual Xeon processor) with 32GB of memory. A total of 500,000 9kB buffers were allocated to Virtuoso. For benchmarking purposes, we tested both direct and reverse queries on all eligible probes.

5 Results

Although all 19 hierarchies of SNOMED CT were processed, due to space constraints, we only report results on the 7 largest and most clinically relevant hierarchies, covering 84% of all concepts. Table 1 provides summary statistics of our analysis of the lattice-theoretic property of SNOMED CT by the "direct" approach (Section 3.1). Table 2 contains the corresponding results for the "reverse" approach (Section 3.3). The first column contains names of the hierarchies and their SNOMED CT ID numbers. The second column (Size) displays the number of concept nodes for each hierarchy. The PP column represents the total number of probe pairs (see Definition 1). The NL column is the total number of non-lattice pairs found. PP% is the percentage of probe pairs among all pairs, NL% is the percentage of non-lattice pairs among all probe pairs, i.e. NL/PP. AT(ms) is the average query time for all probe pairs in milliseconds and TT(h) is the total query time for all probe pairs in hours.

The SPARQL implementation of *NuMi* was run on each probe, whose total number for each hierarchy is displayed in the PP column. Over 1.6 billion queries were issued to Virtuoso for testing the probes. Note that the proportion of probe pairs (PP%) and non-lattice pairs (NL%) is significantly lower in the reverse approach compared to the direct approach. Overall, about 1.9% of all pairs are non-lattice pairs with the direct approach, but only 0.009% are non-lattice pairs with the reverse approach.

Table 1. Summary of query results (direct approach)

Hierarchy	Size	PP	NL	PP%	NL%	AT(ms)	TT(h)
Body Structure (123037004)	31,309	84,809,733	57,134,031	17.3	67.4	13.463	317.16
Clinical Finding (404684003)	101,027	871,780,229	53,988,609	17.1	6.2	6.758	1,636.53
Organism (410607006)	30,149	682,168	74,460	0.2	10.9	2.657	0.50
Pharmaceutical (373873005)	16,718	24,204,248	1,034,394	17.3	4.3	2.569	17.27
Procedure (71388002)	55,328	307,032,527	35,802,137	20.1	11.7	9.950	848.60
Specimen (123038009)	1,209	227,941	28,464	32.1	12.5	2.657	0.17
Substance (105590001)	23,514	9,695,574	3,291,344	3.5	33.9	3.371	9.08
Total	259,254	1,298,432,420	151,353,439	16.2	11.7	7.84	2,829.31

Table 2. Summary of query results (reverse approach)

Hierarchy	Size	PP	NL	PP%	NL%	AT(ms)	TT(h)
Body Structure (123037004)	31,309	29,934,856	91,787	6.1	0.31	11.710	97.37
Clinical Finding (404684003)	101,027	243,428,748	251,662	4.8	0.1	5.069	342.76
Organism (410607006)	30,149	6,739,818	1,040	1.5	0.02	2.425	4.54
Pharmaceutical (373873005)	16,718	5,190,080	6,446	3.7	0.1	2.020	2.91
Procedure (71388002)	55,328	65,800,235	174,574	4.3	0.26	8.239	150.59
Specimen (123038009)	1,209	53,397	889	7.3	1.66	2.244	0.033
Substance (105590001)	23,514	4,666,656	17,340	1.7	0.4	2.368	3.07
Total	259,254	355,813,790	543,738	4.5	0.2	6.08	601.27

What is more significant is the total number of non-lattice pairs. With the direct approach, a total of 151,353,439 pairs are non-lattice pairs, while "only" 543,738 pairs are non-lattice pairs with the reverse approach (i.e., 0.36% of the pairs in the direct approach). It is also observed that in the direct approach, the size of the set $\mathsf{mub}\{a,b\}$ for probe (a,b) is moderate, ranging from two to a few hundreds. In contrast, the size of the set $\mathsf{mlb}\{x,y\}$ (reverse approach) can reach several thousands.

In summary, the average query time with the direct approach is about the same as with the reverse approach. However, the number of probe pairs to be tested differs significantly between the two approaches (16.2% vs. 4.5% of all possible within each hierarchy). This difference is attributable to the structure of the ontology, not the algorithm. Nonetheless, the reverse approach represents a significant saving by focusing on pairs that do not involve quasi-primes and are not in a hierarchical relationship. The total computational time of about 600 hours is in stark contrast to over 2,800 hours. In practice, domain experts curating the ontology will review either the (non-single) upper bounds resulting from the analysis by the direct approach, or the non-lattice pairs (i.e., pairs with no greatest lower bound) resulting from the analysis by the reverse approach. Therefore, while the reverse approach offers superior computational performance over the direct approach.

6 Discussion

6.1 Technical Significance

To our knowledge this work is the first systematic analysis of the lattice-theoretic properties of a large ontology. Unlike other similar techniques such as Formal Concept

Analysis (FCA), our approach is scalable and can be applied to the entirety of the ontology. In contrast, in their analysis of the same ontology (SNOMED CT) with FCA, Jiang and Chute had to rely on stratified sampling of a limited subset of hierarchies in order to estimate the proportion of anonymous nodes [7].

The availability of efficient tools for Semantic Web technologies including RDF stores and SPARQL query engines primarily enables the integration of large datasets in the framework of the Semantic Web, as illustrated by the increasing amount of linked data available [9]. It also provides alternative implementation options for problems that can be construed as constraints on graphs, including the analysis of lattice-theoretic properties of ontologies. This is particularly important when algorithms for such problems need to be applied to large ontologies, created for real-life applications. While it was somewhat challenging (and not completely intuitive at first) to express the *NuMi* algorithm entirely in a declarative query language such as SPARQL, the alternative, i.e., implementing the algorithm procedurally with a traditional programming language, would have required far more ad hoc coding and have posed different challenges due to the sheer size of SNOMED CT. The only coding required for this work was the scripting needed for choreographing the various elements of our quality assurance pipeline (probe selection, probe testing and limited post-processing of the results for statistical analysis purposes). The bulk of the processing was supported by Virtuoso, the triple store and SPARQL query engine, to which some 2.5 billion queries were sent (counting probe selection and testing).

We also take advantage of the mathematical properties of lattices for optimization purposes. We show that the most intuitive algorithm designed for testing the least upper bond (Section 3.1) is equivalent to the "reverse" algorithm testing the greatest lower bound (Section 3.3). While the complexity of the two algorithms is the same, we show that the reverse algorithm is significantly more efficient due to the structure of the data, reducing total execution time by 80% and reducing the total number of non-lattice pairs by more than 2 orders of magnitude.

6.2 Ontological Significance

Quality assurance in ontologies is an active field for research (see, for example, [22] for quality assurance in biomedical ontologies). The quality of SNOMED CT has been examined from several perspectives.

On the one hand, SNOMED CT is developed using an environment based on description logic. Therefore, all the relations made available through the relational database in which SNOMED CT is distributed are guaranteed to be logically consistent. However, the limited expressiveness of the description logic dialect used by SNOMED CT, $\mathcal{EL}$, severely limits the types of inconsistency discoverable by the DL classifier [14].

Structural approaches have been developed for analyzing SNOMED CT, including the Abstraction Network methodology [17], and have been contrasted with description logics [18]. Examples of errors identified by the Abstraction Network methodology and invisible to the DL classifier include missing IS-A relations and duplicate concepts. The analysis of SNOMED CT using this methodology was restricted to the Specimen hierarchy.

The application of Formal Concept Analysis (FCA) to SNOMED CT was discussed earlier [7]. Its strongest limitation is the lack of scalability. We also showed that our results (based on the lattice-theoretic properties) were generally consistent with that of [7] based on FCA. A benefit of FCA is that, unlike our approach, it provides some explanatory information for missing concepts (anonymous nodes).

Overall, we believe that these various methods provide complementary prospectives on the quality of SNOMED CT and have different strengths and limitations. None of these approaches provides an automated solution to auditing ontologies. In fact, each method identifies deviation from properties assumed to be desirable (e.g., logical consistency in DL, structural properties of lattices). Further analysis of the consequences of such deviation generally requires manual review by domain experts. These approaches, however, are important elements of quality assurance as they point out potential problems and help focus the review of the ontology. In addition to scalability, one advantage of our approach is applicable to underspecified ontologies consisting mostly of a taxonomic backbone, while the other three approaches require a richer set of source data (e.g., associative relations across hierarchies) for maximal efficiency.

6.3 Biomedical Significance

Using our approach, we have been able to reproduce the findings of Jiang and Chute [7], which we have discussed in [19]. As shown through the example of the procedure concept "Hypophysectomy" used in [7], the presence of anonymous nodes revealed by FCA generally corresponds to the presence of non-lattice fragments in the ancestors of this concept. The trade-off between the two approaches is in part more explanatory power (FCA) vs. scalability to the entire ontology.

One unresolved question is the extent to which the concepts identified as "missing" (anonymous nodes in FCA, missing least upper bound in the lattice analysis) have clinical utility, i.e., whether their absence from SNOMED CT is detrimental to some of its uses, including clinical documentation and clinical decision support. With over 300,000 concepts, SNOMED CT is the largest clinical ontology currently available and both its developers and the user community are reluctant to increase its size unless this is really needed to satisfy some use case.

SNOMED CT can be extended through post-coordination, i.e., by refining existing concepts and by combining existing concepts in a controlled way. For example, while anatomical structures are lateralized in SNOMED CT (e.g., "Left kidney"), procedure concepts are generally not lateralized (e.g., "Nephrectomy", but no "Left nephrectomy"). However, lateralized procedure concepts can be created by refining the non-lateralized procedure concept with a lateralized anatomical structure (e.g., creating "Left nephrectomy" by making "Left kidney" the procedure site of "Nephrectomy"). Similar concepts could be created through post-coordination for variants of nephrectomy (e.g., "Partial nephrectomy" and "Cadaver nephrectomy").

While these concepts could also be created in SNOMED CT as pre-coordinated concepts, creating concepts for all possible combinations of variants would likely result in combinatorial explosion and management issues for both developers and users. Therefore, it is unclear whether the concepts identified as missing from a structural perspective have enough clinical utility (defined by the editorial guides for SNOMED CT) to warrant their creation as pre-coordinated concepts.

It is beyond the scope of this paper to assess clinical utility. Our study simply identifies areas from which concepts are potentially missing. The availability of models (not just guidelines) for desirable levels of pre-coordination (vs. excessive pre-coordination) would enable us to filter out cases were the absence of a least upper bound corresponds to a feature in SNOMED CT, rather than a bug.

6.4 Limitations, Generalization and Future Work

As mentioned earlier, our algorithm can help guide the manual curation of SNOMED CT by domain experts, but is incomplete by itself for quality assurance purposes. In particular, our algorithm is known to identify false positives, i.e., concepts missing from a the structural perspective of lattices, but lacking clinical utility for them to be created as pre-coordinated concepts in SNOMED CT. In future work, we will work on the formalization of criteria for excessive pre-coordination and filter out these cases from the output of our algorithm. It would be desirable to develop computable metrics to assess the quality of ontological systems.

The strengths of our approach and FCA could be combined. Our approach could be used for efficient identification of non-lattice pairs, and followed by a limited analysis of the local fragments around the non-lattice pairs by FCA. The combined methods are expected to provide a more efficient and powerful pipeline for quality assurance of SNOMED CT. This combination is also a possible direction for future work.

Unlike FCA and other structural methodologies, our approach relies solely on the taxonomic backbone of ontologies and could therefore be applied to other ontologies, including ontologies lacking a rich network of associative relations (e.g., the Gene Ontology [4]).

Our results summarized in Table 1 and Table 2 were carried out by running SPARQL queries coding *NuMi* and instantiated for all probe pairs. The queries can be parallelized and run independently by partitioning the probe pairs into smaller groups. If we spread the queries to n processors, each with its own local, independent SNOMED CT RDF triple store, then an n-fold reduction of the computational time can be achieved.

Finally, we also plan to extend our analysis to other order-theoretic properties such as “part of”, which is a key relation in anatomical ontologies such as the Foundational Model of Anatomy [2]. This may suggest the formulation of other order-theoretic properties in SPARQL than the lattice-theoretic one presented in this study.

Acknowledgements

This research was supported in part by the Intramural Research Program of the National Library of Medicine (NLM), and in part by Clinical and Translational Science Award (UL1 RR024989) from the National Center for Research Resources (NCRR) and Comprehensive Cancer Center Support Grant (5P30 CA043703-16) from National Cancer Institute (NCI), all of the National Institutes of Health (NIH).

References

1. Donnelly, K.: SNOMED-CT: The advanced terminology and coding system for eHealth. Stud. Health Technol. Inform. 121, 279–290 (2006)
2. FMA, `http://sig.biostr.washington.edu/projects/fm/`
3. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Heidelberg (1999)
4. Gene Ontology, `http://www.geneontology.org/`
5. Gierz, G., Hofmann, K.H., Keimel, K., Lawson, D.J., Mislove, M., Scott, D.S.: Continuous Lattices and Domains. Encyclopedia of Mathematics and its Applications, vol. 93. Cambridge University Press, Cambridge (2003)
6. International Health Terminology Standard Development Organization (IHTSDO), `http://www.ihtsdo.org/`
7. Jiang, G., Chute, C.G.: Auditing the semantic completeness of SNOMED CT using formal concept analysis. J. Am Med. Inform. Assoc. 16(1), 89–102 (2009)
8. Joslyn, C.: Poset Ontologies and Concept Lattices as Semantic Hierarchies. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 287–302. Springer, Heidelberg (2004)
9. Linked data, `http://linkeddata.org/`
10. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. J. Exp. Theor. Artif. Intell. 14(2-3), 189–216 (2002)
11. Priss, U.: Formal Concept Analysis as a Tool for Linguistic Data Exploration. In: Hitzler, P., Scharfe, H. (eds.) Conceptual Structures in Practice. Chapman & Hall/CRC Studies in Informatics Series, pp. 177–198 (2009)
12. RDF, `http://www.w3.org/RDF/`
13. SPARQL, `http://www.w3.org/TR/rdf-sparql-query/`
14. Suntisrivaraporn, B., Baader, F., Schulz, S., Spackman, K.: Replacing SEP-Triplets in SNOMED CT using tractable description logic operators. In: Bellazzi, R., Abu-Hanna, A., Hunter, J. (eds.) AIME 2007. LNCS (LNAI), vol. 4594, pp. 287–291. Springer, Heidelberg (2007)
15. Troy, A., Zhang, G.Q., Tian, Y.: Faster concept analysis. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 206–219. Springer, Heidelberg (2007)
16. Virtuoso, `http://virtuoso.openlinksw.com/`
17. Wang, Y., Halper, M., Min, H., Perl, Y., Chen, Y., Spackman, K.A.: Structural methodologies for auditing SNOMED. J. Biomed. Inform. 40(5), 561–581 (2007)
18. Wei, D., Bodenreider, O.: Using the Abstraction Network in complement to Description Logics for quality assurance in biomedical terminologies - A case study in SNOMED CT Medinfo 2010 (in press, 2010)
19. Zhang G.Q and Bodenreider O. Large-scale, exhaustive lattice-based structural auditing of SNOMED CT. In: American Medical Informatics Association (AMIA) Fall 2010 Symposium (in press, 2010)
20. Zhang, G.Q.: Logic of Domains. Birkhäuser, Basel (1991)
21. Zhang, G.Q.: Quasi-prime algebraic domains. Theoretical Computer Science 155, 221–264 (1996)
22. Zhu, X., Wei, J.W., Baorto, D., Weng, C., Cimino, J.: A review of auditing methods applied to the content of controlled biomedical terminologies. J. Biomedical Informatics 42, 412–425 (2009)
23. Zweigenbaum, P., Bachimont, B., Bouaud, J., Charlet, J., Boisvieux, J.F.: Issues in the structuring and acquisition of an ontology for medical language understanding. Methods Inf. Med. 34(1-2), 15–24 (1995)