Proceedings of the IEEE International Conference on Big Data 2014 (IEEE BigData 2014) 2014:754-759.

MaPLE: A MapReduce Pipeline for Lattice-based Evaluation and Its Application to SNOMED CT

Guo-Qiang Zhang*, Wei Zhu*, Mengmeng Sun*, Shiqiang Tao*, Olivier Bodenreider†, Licong Cui*
*Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH 44106
†National Library of Medicine, Bethesda, MD 20892, USA

***Abstract*—Non-lattice fragments are often indicative of structural anomalies in ontological systems and, as such, represent possible areas of focus for subsequent quality assurance work. However, extracting the non-lattice fragments in large ontological systems is computationally expensive if not prohibitive, using a traditional sequential approach. In this paper we present a general MapReduce pipeline, called MaPLE (MapReduce Pipeline for Lattice-based Evaluation), for extracting non-lattice fragments in large partially ordered sets and demonstrate its applicability in ontology quality assurance. Using MaPLE in a 30-node Hadoop local cloud, we systematically extracted non-lattice fragments in 8 SNOMED CT versions from 2009 to 2014 (each containing over 300k concepts), with an average total computing time of less than 3 hours per version. With dramatically reduced time, MaPLE makes it feasible not only to perform exhaustive structural analysis of large ontological hierarchies, but also to systematically track structural changes between versions. Our change analysis showed that the average change rates on the non-lattice pairs are up to 38.6 times higher than the change rates of the background structure (concept nodes). This demonstrates that fragments around non-lattice pairs exhibit significantly higher rates of change in the process of ontological evolution.**

I. Introduction

Ontologies represent not only the concepts, but just as importantly, the relationships between the concepts. Ontologies have become a critical component in informatics and data intensive applications. They are used, for example, to handle terminological heterogeneity, facilitate system interoperability and integration, enable knowledge discovery [1], [2], [3], [4], and manage biomedical big data [5], [6].

However, ontological systems are often incomplete, under-specified, and non-static. New applications call for new ontologies or expansion and enhancement of existing ones. Many additional factors, such as merging or reusing existing ontologies and porting to a common representation framework, may introduce inconsistencies and unintended artifacts. Thus Ontology Quality Assurance (OQA) is an indispensable part of the ontological engineering lifecycle [7].

Due to the increasingly large size and structural complexity of modern ontologies, particularly those from biomedicine, OQA has been hampered by the lack of systematic and scalable methods necessary to keep pace with the evolution and emergence of ontological systems. The state of affairs is that [8]: "given modern tooling and computer power, the barriers for quality assurance can now be overcome, although no well-integrated toolset is yet available."

One of the generally applicable ontology design principle or pattern is that the subsumption relationship (IS-A hierarchy) should form a lattice [9]. However, extracting non-lattice fragments in large ontological systems is computationally expensive if not prohibitive, using a traditional sequential approach. In previous work [9], we analyzed 34 million pairs of SNOMED CT concepts and identified over 518,000 non-lattice pairs using SPARQL queries over an RDF representation of the ontology. However, the time needed for such an exhaustive analysis – 3 months using standard desktop machines – is incompatible with use in real quality assurance applications.

In this paper we present a general MapReduce pipeline, called MaPLE (MapReduce Pipeline for Lattice-based Evaluation), for extracting non-lattice fragments in large partially ordered sets and demonstrate its applicability in ontology quality assurance. MaPLE consists of a sequence of simple order-theoretic and set-theoretic operations designed for mappers and reducers that collectively capture the non-trivial lattice-theoretic property. We show that the lattice-based analysis of hierarchical relations in SNOMED CT, while quadratic in computational complexity, is tractable using MaPLE with Hadoop. After implementing MaPLE in Cloudera Hadoop 4.3 on a 30-node cluster, we systematically extracted non-lattice fragments in 8 SNOMED CT versions from 2009 to 2014, with an average total compute time of less than 3 hours per version. With dramatically improved turn-around time for analysis, 3 hours instead of 3 months, MaPLE makes it feasible not only to perform exhaustive structural analysis of large ontological hierarchies, but also to systematically track structural changes between versions.

The rest of the paper is organized as follows. In Section II we recall background information on related topics. In Section III we describe our MapReduce algorithm MaPLE for extracting non-lattice fragments in a partially ordered set. In Section IV we describe the implementation of MaPLE in a Hadoop local cloud, and an application for extracting all non-lattice fragments in SNOMED CT. In Section V we present the main statistics and results about non-lattice fragments in SNOMED CT versions and a performance evaluation of MaPLE. The last section contains conclusive remarks.

II. Background

A. SNOMED CT

Developed by the International Health Terminology Standard Development Organization (IHTSDO), SNOMED CT is the world's largest clinical terminology and provides broad coverage of clinical medicine, including findings, diseases, and procedures for use in electronic medical records. The international release of SNOMED CT is produced twice a year, in January and in July, reflecting both changes to medical

knowledge (e.g., new drugs) and changes to the editorial process (e.g., changes to the representation of anatomical entities). In addition, the member countries of the IHTSDO also create extensions of SNOMED CT, with additional concepts specific to the needs of a particular country. For example, the U.S. extension of SNOMED CT is now packaged together with the international release and released in March and September.

From a structural perspective, SNOMED CT can be seen as a series of large directed acyclic graphs, one for each of its 19 "sub-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 sub-hierarchies except for the root. Each concept comes with a SNOMED CT identifier, which is an integer. SNOMED CT concepts are linked by hierarchical relations within each sub-hierarchy, such as "Tissue specimen from heart" IS-A "Tissue specimen."

B. Lattice and Formal Concept Analysis

We recall some basic definitions and notations in partial orders and lattice theory [10], [11].

In a partially ordered set (poset) L, an element u is called an *upper bound* of a subset $X \subseteq L$, if for each $x \in X$ we have $x \leq u$. An element m is called a *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 of X. The notions of lower bound, maximal lower bound, greatest lower bound, are 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 least upper bound (join) and greatest lower bound (meet). The meet and join of binary sets are often 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 $\bigwedge S$ (meet).

In connection with ontology, one can think of concepts as elements of a poset, and the ordering relation as the subsumption relation [13]. 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 least common ancestor of x and y, and the meet $x \wedge y$ is their greatest common descendant.

Formal Concept Analysis (FCA) [14] is a general lattice-based method for extracting higher-level organizational information from lower-level classification of objects according to their attributes. The primary notion in FCA is a *formal context* (or context), (O, A, R), where O is a collection of objects, A a collection of attributes, and R is a relation $R \subseteq O \times A$. A fundamental theorem of FCA is that every formal context determines a complete lattice, and reversely, every complete lattice can be derived from a formal context.

C. Lattice-based Ontology Auditing

Well-formed ontologies often have a lattice structure [9]. 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 with FCA [12]. 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 obtains a complete lattice automatically using FCA [14].

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 may not have been crystalized. This may reveal gaps in conceptual modeling.

Though desirable, the lattice property of ontologies is not always found in most biomedical ontologies. Fig. 1 is a non-lattice fragment [9] from SNOMED CT. "Tissue specimen from trunk" is not a concept in SNOMED CT. Including it appropriately makes this fragment lattice conforming.

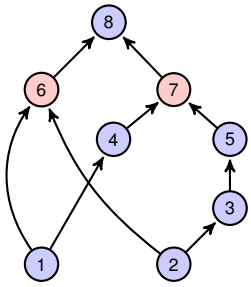

Fig. 1. A non-lattice fragment in SNOMED CT. 1: Tissue specimen from breast; 2: Tissue specimen from heart; 3: Specimen from heart; 4: Specimen from breast; 5: Specimen from mediastinum; 6: Tissue specimen; 7: Specimen from trunk; 8: Specimen.

D. MapReduce

MapReduce [15] is a distributed programming environment to process large amounts of data in a scalable way. A MapReduce job consists of a mapper and a reducer function, specified by the user to process data in the form of key-value pairs. Such a job is automatically broken into tasks executed in parallel across a cluster of machines called compute nodes. Designing efficient algorithms in MapReduce requires a different mentality than the transitional approach, with more attention paid to data locality, job break-down, and a trade-off between parallelism and communication latency.

III. The MaPLE Algorithm

In this section, we introduce the algorithmic aspect of MaPLE, our MapReduce Pipeline for Lattice-based Evaluation. It takes a partially ordered set (poset) as input, and generates the collection of all non-lattice pairs in the poset.

A. Non-lattice Pairs

A pair of elements a, b in a poset $(L, \leq)$ is a non-lattice pair if $|\ \mathsf{mub}\{a, b\}\ | > 1$, i.e. the pair a, b has at least two minimal upper bounds. For example, in the poset given in Fig. 1, we have $\mathsf{mub}\{1, 2\} = \{6, 7\}$, so the size of $\mathsf{mub}\{1, 2\}$ is 2, making $(1, 2)$ a non-lattice pair in this poset. On the other hand, we have $\mathsf{mub}\{1, 3\} = \{7\}$, and so $(1, 3)$ is a lattice pair.

Thus, finding non-lattice pairs reduces to computing the minimal upper bounds, achieved through a sequence of set-theoretic operations using closures as explained next.

B. Computing Minimal Upper Bounds

Given a set X of elements in a poset L, we use $\Uparrow X$ to denote the set of all common ancestors of X, i.e., $\Uparrow X := \{a \mid \forall x \in X, x < a\}$. Note that $X \cap (\Uparrow X)$ is always empty. When X is a singleton, i.e., $X = \{x\}$, we write $\Uparrow x$ for $\Uparrow\{x\}$. Similarly, we define $\uparrow X := \{a \mid \exists x \in X, x < a\}$. Thus, $\uparrow X$ represents its strict upper closure. It is straightforward to check that for any set $X, Y \subseteq L$,

$$\Uparrow X = \bigcap_{a \in X} \Uparrow a, \quad \text{and} \quad \uparrow Y = \bigcup_{b \in Y} \uparrow b.$$

For singletons, we have $\Uparrow x = \uparrow x$. Two elements x, y are called *incomparable* if neither $x \leq y$, nor $y \leq x$.

The correctness of our algorithm for computing minimal upper bounds rests on the following simple result (Theorem 1). Before introducing this result, the following lemma should be helpful (whose proof is omitted):

Lemma 1: Let x, y be elements in a poset L. If $y \leq x$ but $x \notin \uparrow y$, then $x = y$.

Theorem 1: Let X be a set of pairwise incomparable elements of a poset L, with the size of X at least two. We have

$$\mathsf{mub}(X) = \Uparrow X - \uparrow(\Uparrow X).$$

Proof: Let $m \in \Uparrow X - \uparrow(\Uparrow X)$. Since $m \in \Uparrow X$, m is an upper bound of X. Since $m \notin \uparrow(\Uparrow X)$, we have $m \notin \uparrow t$ for each $t \in \Uparrow X$.

For any upper bound u of X (i.e., $x \leq u$ for each $x \in X$) with $u \leq m$, we need to show that $u = m$. Since u is an upper bound of X, which contains at least two incomparable elements, we have $u \in \Uparrow X$. Also, because $m \notin \uparrow t$ for each $t \in \Uparrow X$, we have $m \notin \uparrow u$. By Lemma 1, $m = u$. Hence, m is a minimal upper bound of X. ∎

A special case for Theorem 1 is a pair of incomparable elements x and y, which is stated as:

$$\mathsf{mub}(\{x, y\}) = (\uparrow x \cap \uparrow y) - \bigcup_{t \in \uparrow x \cap \uparrow y} \uparrow t. \tag{1}$$

Fig. 2 displays a general algorithm to detect non-lattice pairs in a poset. The input (line 1) is a poset L, represented by its associated set of pairs x, y with $x \leq y$. The output (line 2) is a set of detected non-lattice pairs and their minimal upper bounds. Lines 3-5 collect ancestors for each element of L. Lines 6-17 detect non-lattice pairs by computing minimal upper bounds using Equation (1) for each candidate pair, and output pairs with more than one minimal upper bound.

IV. MaPLE Implementation for SNOMED CT

The algorithm (Fig. 2) to detect non-lattice fragments is implemented using two MapReduce jobs (Fig. 3 and Fig. 4), for exhaustive, structural analysis of SNOMED CT.

Since finding the ancestors of a concept is a basic operation repeatedly performed throughout the MaPLE algorithm, we assume that the transitive closure of the SNOMED CT concepts hierarchy has been precomputed, to save overall time.

```
1: Input: A poset L
2: Output: A set R of non-lattice pairs and their minimal upper bounds
3: for each element x ∈ L do
4:     Find the set of its ancestors ↑x
5: end for
6: Initialize R = ∅
7: for each pair of elements (x, y) do
8:     Calculate ↑x ∩ ↑y
9:     Initialize U ← ∅
10:    for each t ∈ ↑x ∩ ↑y do
11:        U ← U ∪ ↑t
12:    end for
13:    if x and y are incomparable and |↑x ∩ ↑y − U| > 1 then
14:        R ← R ∪ ((x, y), ↑x ∩ ↑y − U)
15:    end if
16: end for
17: return R
```

Fig. 2. Algorithm for detecting non-lattice pairs in a poset.

A. MapReduce Jobs

Given a set of transitively closed pairs of SNOMED CT concepts, the first MapReduce job described in Fig. 3 simply collects ancestors for each concept. In the mapping phase (lines 3-5), each mapper reads in a set of transitively closed concept pairs and emits key-value pairs (c, a) where c is a concept and a is an ancestor of c. In the reduce stage (lines 6-8), each reducer collects all the ancestors $\uparrow c = \{a_1, a_2, \ldots\}$ of a concept c and emits concept-ancestors pairs $(c, \uparrow c)$.

```
1: Input: Transitive closure concept pairs
2: Output: Concept-ancestors pairs
3: class Mapper
4:     method Map(concept c, ancestor a)
5:         Emit(c, a)
6: class Reducer
7:     method Reduce(c, {a1, a2, ...})
8:         Emit(c, {a1, a2, ...})
```

Fig. 3. MapReduce Job 1: collecting ancestors for each concept.

Fig. 4 contains the description of the second MapReduce job. The input is a set of concept-ancestors pairs $(c, \uparrow c)$ resulting from MapReduce Job 1. In implementation, such input concept-ancestors pairs are not only split and fed into multiple mappers, but also read by all the mappers and reducers as *DistributedCache* in Hadoop.

In the map stage, each mapper first loads all concept-ancestors pairs $(c, \uparrow c)$ into a hash map CA to facilitate the generation of candidate pairs (lines 4-7). Then each mapper generates candidate pairs for each concept c_1 (lines 8-13) by iterating through each concept c_2 in CA, checking if c_1's concept identifier is less than c_2's concept identifier to ensure uniqueness of the pair. A concept pair (c_1, c_2) is then emitted as a key and their ancestors $(\uparrow c_1, \uparrow c_2)$ where $\uparrow c_1 = \{a_{11}, a_{12}, \ldots\}$, $\uparrow c_2 = \{a_{21}, a_{22}, \ldots\}$, respectively, as a value. The requirement that c_1's concept identifier is less than c_2's concept identifier avoids the situation that the same concept pair generates two different keys.

In the reduce stage, each reducer first loads all concept-ancestors pairs $(c, \uparrow c)$ into a hash map CA (lines 15-18). Then each reducer checks if c_1 and c_2 in a concept pair

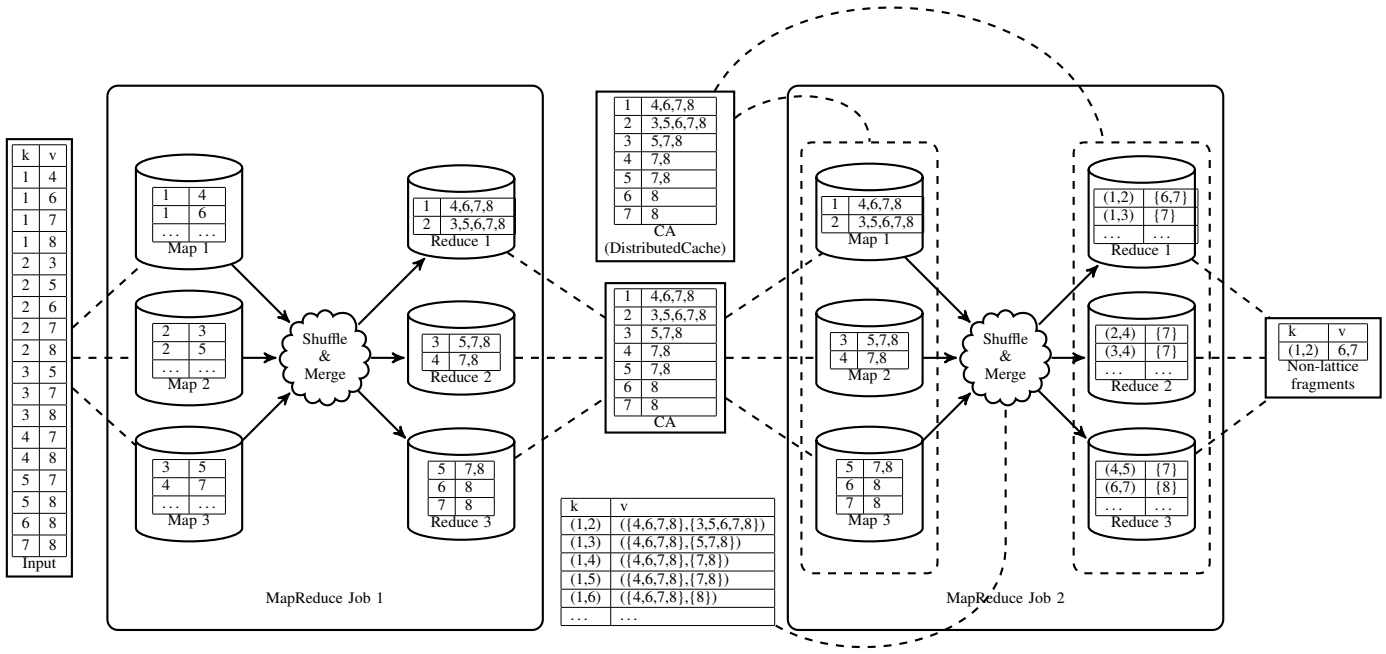

Fig. 5. Illustrative diagram of data flow for MapReduce Job 1 and MapReduce Job 2, with input poset being the one given in Fig. 1. Key-value pair transformations are as follows: $(c,a) \longrightarrow (c, {\uparrow}c)$ for Job 1, and $((c_1,c_2),({\uparrow}c_1,{\uparrow}c_2)) \longrightarrow ((c_1,c_2),({\uparrow}c_1 \cap {\uparrow}c_2) - (\bigcup_{t \in {\uparrow}c_1 \cap {\uparrow}c_2} {\uparrow}t))$ for Job 2. In the beginning of Job 2, a copy of hashed CA is cached for all compute nodes, and another copy of CA serves as input for pairing. Dotted edges stand for data feed or distributed cache.

are incomparable (lines 20-21), calculates their minimal upper bounds (lines 22-27), and emits pairs with more than one minimal upper bound as keys and their minimal upper bounds as values (lines 28-30).

Fig. 5 illustrates the two MapReduce jobs in the MaPLE pipeline using the poset example given in Fig. 1.

B. Optimization Strategies

To reduce the total number of input and resulting pairs yet without missing any non-lattice fragments, we incorporate two optimization strategies in MaPLE implementation, which were first introduced in [12].

1) Leveraging Duality and Reversing Order: After obtaining the transitive closure of a set of SNOMED CT pairs $\{(c,a)\}$, we reverse the pairs and take the pairs $\{(a,c)\}$ as input for MaPLE. Ontological hierarchies have a "fan-out" shape: they tend to have fewer upper level concepts representing more general entities, while having lower level concepts representing more specific entities. Since lattices can be viewed either top-down or bottom-up, only one direction needs to be tested for lattice property. The effect of this is that instead of computing minimal upper bounds, we compute the maximal lower bounds in the original hierarchy.

2) Concepts with a Unique Parent: The second optimization strategy is skipping concepts with a unique parent in the map stage of MapReduce Job 2. The rationale for considering concepts with a single parent is that whenever such a concept is involved in a non-lattice pair, there must be an ancestor of this concept already involved in a non-lattice fragment [12], [9]. In implementation, all concepts with a unique parent are extracted and loaded as a *DistributedCache* in the map stage of MapReduce Job 2 (Fig. 4).

C. Experimental Environment

We run MaPLE on a private cloud using Cloudera Hadoop 4.3 (Hadoop 2.0.0-cdh4.3.0) with 1 master node and 30 slave nodes. The master node uses a dual quad-core Intel Xeon 5150 2.66GHz, while the compute nodes are machines with dual quad-core Intel Xeon 5450 3.0GHz processors, all running RedHat Enterprise Linux 6.4. Each node has a local 146GB SAS hard drive. Each server has 16GB of RAM. The compute nodes are interconnected using a low-latency 10 Gigabit Ethernet network based on Arista switches and Intel NetEffects Ethernet adapters.

V. Results

A. Basic SNOMED CT Statistics and Non-lattice Pairs

We ran MaPLE on 8 versions of SNOMED CT from 2009 to 2014, dated 07/2009 (i.e., July 2009), 01/2010, 01/2011, 01/2012, 07/2012, 01/2013, 07/2013, and 03/2014. Table I summarizes the basic results about each version of SNOMED CT. The 07/2009 version contained a total of 306,627 concepts, with 445,549 direct IS-A relationship connecting concepts. Among all possible concept pairs, 559,182 were found to be non-lattice pairs. The MaPLE processing of all 07/2009 SNOMED CT hierarchies in the 30-node Cloudera Hadoop environment took a total 10,168 seconds, roughly 2.8 hours.

Table II summarizes the results for the 10 largest sub-hierarchies in the most recent (03/2014) version. For the

	07/2009	01/2010	01/2011	01/2012	07/2012	01/2013	07/2013	03/2014
Total Number of Concepts	306,627	290,078	292,405	294,797	295,311	296,876	297,695	299,286
Total Number of IS-A Relations	445,549	430,489	435,294	438,711	438,927	441,589	443,796	445,357
Total Number of Non-lattice Pairs	559,182	566,239	576,688	568,535	581,754	574,204	584,934	586,771
Compute Time (s)	10,168	74,169	6,726	7,822	7,883	8,176	8,430	11,166

TABLE I. SUMMARY OF THE BASIC STATISTICS USING MAPLE TO PROCESS 8 SNOMED CT VERSIONS SINCE 2009.

Sub-Hierarchy (SNOMED Identifier)	N	TC	NL	PP	NL%	T(s)
Body Structure (123037004)	30,623	937,549	90,465	468,868,753	0.01929	738
Clinical Finding (404684003)	100,652	1,937,520	277,071	5,065,362,226	0.00547	7,875
Observable Entity (363787002)	8,314	49,006	613	34,557,141	0.00177	49
Organism (410607006)	33,175	380,364	1,530	550,273,725	0.00028	165
Pharmaceutical/Biologic Product (373873005)	16,797	123,940	6,819	141,061,206	0.00483	71
Physical Object (260787004))	4,538	31,897	364	10,294,453	0.00354	39
Procedure (71388002)	54,091	997,984	189,110	1,462,891,095	0.01293	1,768
Qualifier Value (362981000)	8,978	39,863	380	40,297,753	0.00094	41
Social Context (48176007)	4,824	33,122	804	11,633,076	0.00691	36
Substance (105590001)	23,962	281,550	18,019	287,076,741	0.00063	107

TABLE II. SUMMARY OF THE RESULTS FOR THE 03/2014 VERSION OF SNOMED CT. N - THE TOTAL NUMBER OF CONCEPTS IN THE SUB-HIERARCHY; TC - THE TOTAL NUMBER OF TRANSITIVE-CLOSED PAIRS IN THE SUB-HIERARCHY; NL - THE TOTAL NUMBER OF NON-LATTICE PAIRS IN THE SUB-HIERARCHY; PP - THE TOTAL NUMBER OF ALL POSSIBLE PAIRS OBTAINED BY THE FORMULA $n \times (n-1)/2$, WHERE n IS THE NUMBER OF CONCEPT IN THE SUB-HIERARCHY; NL% - THE PERCENTAGE OF NON-LATTICE PAIRS AMONG ALL POSSIBLE PROBE PAIRS, I.E. NL/PP; T(S) - THE MAPLE COMPUTE TIME IN SECONDS IN OUR HPCC ENVIRONMENT.

```
1: Input: Concept-ancestors pairs
2: Output: Non-lattice pairs and their minimal upper bounds

3: class Mapper
4:     method Initialize
5:         Initialize a HashMap CA to load concept-ancestors pairs

6:     method Setup
7:         Load Distributed Cache files and update CA

8:     method Map(c1, {a11, a12, ...})
9:         for each concept c2 in CA do
10:            if c1.ID < c2.ID then          ▷ c1's ID is less than c2's ID
11:                Emit((c1, c2), ({a11, a12, ...}, {a21, a22, ...}))
12:            end if
13:        end for

14: class Reducer
15:    method Initialize
16:        Initialize a HashMap CA to load concept-ancestors pairs

17:    method Setup
18:        Load Distributed Cache file and update CA

19:    method Reduce((c1, c2), {({a11, a12, ...}, {a21, a22, ...})})
20:        N ← {a11, a12, ...} ∪ {a21, a22, ...}          ▷ Union ancestors
21:        if c1 ∉ N and c2 ∉ N then          ▷ c1, c2 are incomparable
22:            A ← {a11, a12, ...} ∩ {a21, a22, ...}      ▷ Intersect ancestors
23:            U ← ∅
24:            for each concept a in A do
25:                U ← U ∪ CA.Get(a)          ▷ Union a's ancestors
26:            end for
27:            B ← A − U          ▷ Calculate minimal upper bounds
28:            if |B| > 1 then
29:                Emit((c1, c2), B)
30:            end if
31:        end if
```

Fig. 4. MapReduce Job 2: detecting non-lattice fragments.

sub-hierarchy "Body Structure," there were 30,623 concepts and 937,549 transitive closure pairs. The number of non-lattice pairs found was 90,465, which is about 0.019% of all 468,868,753 possible pairs. It can also be seen that processing most of the sub-hierarchies took just one or two minutes. Most of the time was spent on processing larger hierarchies. For example, processing "Clinical Finding," the largest sub-hierarchy, took about 2 hours. The total computing time for the 03/2014 version was 11,166 seconds, around 3 hours.

B. Comparison of SNOMED CT Versions

MaPLE allows us to track the changes of non-lattice pairs between different versions. Fig. 6 shows the summary of the differences between SNOMED CT versions using the percentage of non-lattice pair changes. We use |N-O|/|N| and |O-N|/|O| to show the difference. Here |X| stands for the size of the set X. Therefore, |N-O|/|N| is the percentage of non-lattice pairs which is newly added to the later version. |O-N|/|O| is the percentage of non-lattice pairs which is removed from the previous version. It is important to note that |N-O|≠|N|-|O| in general – we can have two equal-sized sets that have nothing in common, and so in the extreme, |N-O|=|N| but |N|-|O|=0.

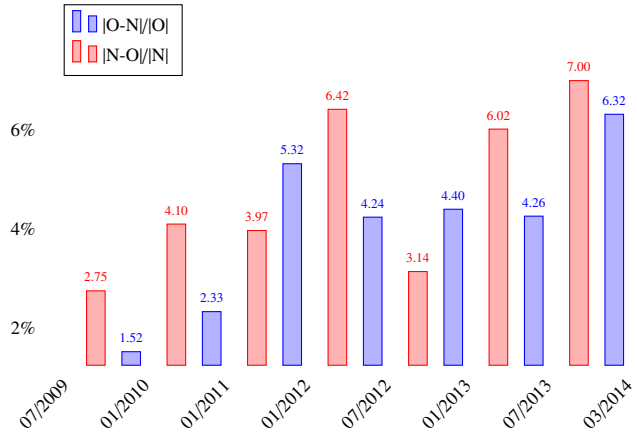

Fig. 6. Percentage changes of non-lattice pairs in 8 SNOMED CT versions.

The colored bars in Fig. 6 show the percentage changes between versions of SNOMED CT. Comparing the 01/2010 version with the 07/2009 version, we found 15,582 non-lattice pairs in the 01/2010 version but not in the 07/2009 version (N-O). This represented 2.75% (|N-O|/|N|) of the total number

of non-lattice pairs in the 01/2010 version. On the other hand, 8,525 non-lattice pairs were found in the 07/2009 version but not 01/2010 (O-N), which amounted to 1.52% of the non-lattice pairs in the 07/2009 version. Similarly, for the 01/2011 version, 13,210 (2.33%) non-lattice pairs were excluded from the earlier 01/2010 version and 23,659 (4.10%) non-lattice pairs were included. One can see that the largest percentage change took place in the most recent (03/2014) version, which contained 41,103 new non-lattice pairs.

Our results show that the change rates in non-lattice pairs are significantly higher than the change rates of the corresponding underlying structure (nodes and IS-A edges) between the 8 SNOMED CT versions. With respect to concept nodes, the average percentage change rate in non-lattice pairs is 38.6 times higher for "in old but not in new" (O-N) and 8.4 times higher for "in new not in old" (N-O). With respect to IS-A edges, the average percentage change rate in non-lattice pairs is 4.4 times higher for (O-N) and 3.5 times higher for (N-O). This suggests that fragments around non-lattice pairs exhibit significantly higher rates of change in ontological evolution.

Scalability is one of the most important features of cloud computing. We performed an experiment on the number of reducer tasks in MapReduce Job 2 for MaPLE, a more computationally intensive step. We found a linear decrease of the computing time when the number of reducer tasks ranged from 1 to 10. However, for 15 to 60 reducer tasks, we see a diminishing return in saved time. This may be due to the fact that when splitting intermediate results from mapper into a large number of smaller file fragments for reducer, the task setup and scheduling time may become a more significant contributor in total time. For the computing time reported in Table I, 9 reducer tasks were involved.

VI. Conclusion

In this paper we introduced a general MapReduce pipeline, called MaPLE for extracting non-lattice fragments in large partially ordered sets and demonstrate its applicability in ontology quality assurance. Our contributions include: a general MapReduce lattice-checking algorithm for posets; an application of MaPLE for exhaustive analysis of 8 SNOMED CT versions since 2009; and a global change analysis of such SNOMED CT versions with respect to the underlying concepts, relationships, and non-lattice pairs. Our change analysis indicated that fragments around non-lattice pairs exhibit significantly higher rates of change in the process of ontological evolution.

One limitation of this approach is that it does not take into account the editorial guidelines of the IHTSDO for the development of SNOMED CT, which are used to prevent the creation of unnecessary concepts (from the perspective of clinical utility). If such guidelines were formalized in machine-readable form, we could integrate them into our analysis, in order to prevent certain non-lattice pairs from being seen as errors. In the future, we plan to work with the IHTSDO to tailor our analysis to their quality assurance processes.

Our approach sets the foundation for structural auditing of ontologies in a couple of new directions. One is to mine the large collection of non-lattice fragments to extract clinically interpretable insights in the ontological evolution. The second is to perform large-scale MapReduce analysis of biomedical ontologies taking advantage of the combination of linguistic and structural information conveyed by the ontological systems, along the lines of work reported in [16], [17], [18].

VII. Acknowledgments

This work was supported in part by the Case Western Reserve University CTSA Grant NIH/NCATS UL1TR000439 and by the Intramural Research Program of the NIH, National Library of Medicine. This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University.

References

[1] Bodenreider O. Biomedical ontologies in action: role in knowledge management, data integration and decision support. Geissbuhler A, Kulikowski C, editors. IMIA Yearbook of Medical Informatics 2008. Methods Inf Med 2008;47(Suppl 1):67-79.

[2] Cui L. Ontology-guided health information extraction, organization, and exploration. PhD Dissertation, Department of EECS, Case Western Reserve University, August 2014.

[3] Zhang GQ, Siegler T, Saxman P, Sandberg N, Mueller R, Johnson N, Hunscher D, Arabandi S. VISAGE: A Query Interface for Clinical Research. AMIA Jt Summits Transl Sci Proc. 2010;76-80.

[4] Zhang GQ, Cui L, Lhatoo S, Schuele S, Sahoo S. MEDCIS: Multi-Modality Epilepsy Data Capture and Integration System. AMIA Annu Symp Proc. 2014 (in press).

[5] Jayapandian CP, Chen CH, Dabir A, Lhatoo S, Zhang GQ, Sahoo S. Domain Ontology As Conceptual Model for Big Data Management: Application in Biomedical Informatics. International Conference on Conceptual Modeling 2014 (in press).

[6] Jayapandian CP. Cloudwave: A Cloud Computing Framework for Multi-modal Electrophysiological Big Data. PhD Dissertation, Department of EECS, Case Western Reserve University, August 2014.

[7] Zhu X, Wei JW, Baorto D, Weng C, Cimino J. A review of auditing methods applied to the content of controlled biomedical terminologies. J Biomedical Informatics, Vol. 42, pages 412-25, 2009.

[8] Rector AL, Brandt S, Schneider T. Getting the foot out of the pelvis: modeling problems affecting use of SNOMED CT hierarchies in practical applications. J Am Med Inform Assoc. 2011;18(4):432-40.

[9] Zhang GQ and Bodenreider O. Large-scale, exhaustive lattice-based structural auditing of SNOMED CT. AMIA Annu Symp Proc. 2010;922-26.

[10] Gierz G, Hofmann KH, Keimel K, Lawson DJ, Mislove M, Scott DS. *Continuous Lattices and Domains*. In: Encyclopedia of Mathematics and its Applications (No. 93), Cambridge University Press, 2003.

[11] Zhang GQ. *Logic of Domains*. Birkhäuser, Boston, 1991.

[12] Zhang GQ and Bodenreider O. Using SPARQL to Test for Lattices: application to quality assurance in biomedical ontologies. The Semantic Web-ISWC 2010, pages 273-288.

[13] Joslyn C. Poset ontologies and concept lattices as semantic hierarchies. Lecture Notes in Computer Science, 2004;3127:287-302.

[14] Ganter B, Wille R. Formal Concept Analysis. Springer-Verlag, 1999.

[15] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008;51(1):107-13.

[16] Luo L, Mejino J, Zhang GQ. An Analysis of FMA Using Structural Self-Bisimilarity. J. Biomedical Informatics 2013;46(3):497-505.

[17] Luo L, Xu R, Zhang GQ. Dissecting the Ambiguity of FMA Concept Names Using Taxonomy and Partonomy Structural Information. AMIA Summits Transl Sci Proc. 2013; 2013:157-61.

[18] Zhang GQ, Luo L, Ogbuji C, Joslyn C, Mejino J, Sahoo S. An Analysis of Multi-type Relational Interactions in FMA Using Graph Motifs with Disjointness Constraints. AMIA Annu Symp Proc. 2012;1060-9.