Web-based Interactive Visualization of Non-Lattice Subgraphs (WINS) in SNOMED CT

Wei Zhu, PhD[1], Shiqiang Tao, PhD[2], Licong Cui, PhD[2], Guo-Qiang Zhang, PhD[2]
[1]Case Western Reserve University, Cleveland, Ohio, USA
[2]The University of Texas Health Science Center at Houston, Houston, Texas, USA

Abstract

Non-lattice subgraphs are often indicative of structural anomalies in ontological systems. Visualization of SNOMED CT's non-lattice subgraphs can help make sense of what has been asserted in the hierarchical ("is-a") relation. More importantly, it can demonstrate what has not been asserted, or "is-not-a," using Closed-World Assumption for such subgraphs. A feature-rich web-based interactive graph-visualization engine called WINS is introduced, for supporting non-lattice based analysis of ontological systems such as SNOMED CT. A faceted search interface is designed for querying conjunctively specified non-lattice subgraphs. To manage the large number of possible non-lattice subgraphs, MongoDB is used for storing and processing sets of concepts, relationships, and subgraphs, as well as for query optimization. WINS' interactive visualization interface is implemented in the open source package D3.js. 14 versions of SNOMED CT (US editions from March 2012 to September 2018), with about 170,000 subgraphs in each version, were extracted and imported into WINS. Two types of non-lattice based ontology quality assurance (OQA) tasks were highlighted to demonstrate use cases of WINS in sense-making of such non-lattice subgraphs.

Introduction

One of the generally applicable ontology design principle or pattern is that the subsumption relationship (or *is-a* hierarchy) should form a lattice[1,2]. Non-lattice subgraphs, therefore, are indicative of possible structural anomalies in ontological systems. For example, in SNOMED CT, subgraphs around non-lattice pairs had change rates (from one version to the next) 38 times higher than the change rates of the overall background structure[3]. Hence, visualization of non-lattice subgraphs can support the analysis and sense-making of potential errors and underlying root causes, and to suggest correction measures resulting in the conversion of non-lattice subgraphs to lattice-conforming subgraphs.

Existing browsers such as the SNOMED CT Browser[4] and the NLM SNOMED CT Browser[5] were not designed to support the interactive visualization of non-lattice subgraphs. The size and complexity of SNOMED CT require sophisticated techniques to handle algorithmic, perceptual, and real-time visualization challenges[6]. Hundreds of thousands of non-lattice pairs and non-lattice subgraphs exist in each version of SNOMED CT, making speedy construction of such subgraphs a difficult task. For example, 631,006 non-lattice pairs and 171,011 non-lattice, non-comparable subgraphs were found in the September 2015 version of SNOMED CT (US edition). An efficient searching interface is also indispensable to quickly locate non-lattice subgraphs of interest after they have been generated. Since lattice and non-lattice structures are specific types of directed acyclic graphs (DAGs), classical graph rendering algorithms and techniques need to be adapted and implemented for visualizing subgraphs in ontological systems.

To overcomes such challenges, we introduce a visualization tool called Web-based Interactive Visualization of Non-Lattice Subgraphs (WINS). WINS can effectively display all relevant information around non-lattice subgraphs specified by a user in virtually real-time while allowing the user to perform meaningful interaction with the ontological substructure. This paper serves as a use case for WINS dedicated to supporting non-lattice subgraph visualization for facilitating quality assurance of SNOMED CT.

1 Background

1.1 SNOMED CT

SNOMED CT[7] is the world's largest comprehensive and precise, multilingual premier comprehensive clinical terminology of healthcare-related terminologies worldwide. It provides broad coverage of clinical medicine, including Findings, Diseases, and Procedures to support clinical decision, information retrieval, and semantic interoperability. SNOMED CT is accepted as a common global language for health terms in over 50 countries. In the U.S., the Health Information Technology for Economic and Clinical Health (HITECH) Act[8] have designated SNOMED CT as the U.S.

national terminology standard for multiple purposes including supporting meaningful use of EHRs and cost-effective delivery of care.

1.2 Non-Lattice Based Ontology Quality Assurance

Although being a lattice is a generally accepted ontology design pattern for the *is-a* hierarchy[1,2], this property does not always hold in biomedical ontologies[1,9,10]. Figure 1 shows two examples of structures of SNOMED CT (September 2014 version): lattice structure on the left and non-lattice structure on the right. A *lattice pair* is a pair of concepts that have a unique maximal lower bound and a unique minimal upper bound. For example, on the left, the two concepts (in green) "Myocardial disease" and "Fetal heart disorder" form a lattice pair, since they have a unique minimal upper bound "Heart disease" (in pink) and a unique maximal lower bound "Disorder of fetal myocardium" (in purple). The fragment containing the *lattice pair* is called a lattice subgraph. A *non-lattice pair* (a, b) is defined as a pair of concepts that have at least two maximal lower bounds. For example, on the right, "Heart disease" and "Disorder related to cardiac transplantation" (in green) have two (hence not unique) maximal lower bounds: "Cardiac Transplant disorder" and "Disorder of transplanted heart" (in purple). This fragment (including the maximal lower bounds) is called a non-lattice subgraph.

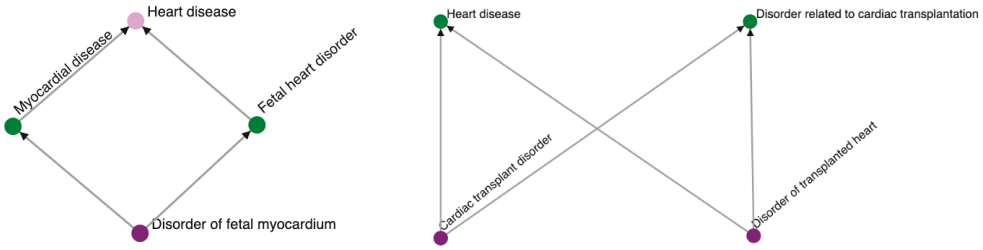

Figure 1: Examples of lattice (left) and non-lattice (right).

1.3 Visualization Tools for SNOMED CT

Ontology visualization is essential to facilitate sophisticated quality assurance work in ontological evolution. Visualization tools for ontologies have been developed for viewing terminology content or hierarchy structure[11,12]. The official *SNOMED Browser*[4] is an online, multilingual, multi-edition ontology browsing application. It enables browsing of International Edition of SNOMED CT, as well as various National Editions. User can enter terms or navigate in the taxonomy hierarchies to find the target concept, with filters in refset or language. The application provides the details of concepts, including descriptions, parents and children, attributes and structural diagrams. *Protégé*[13] is one of the most widely used ontology development tools. Its framework allows interactive creation and visualization of classes in a hierarchical view. Particularly, *Protégé* is extendable via plug-ins, enabling developers to add additional functionality into the *Protégé* system. *Web-Protégé*[14] is a web-based version of *Protégé* that focuses on collaborative ontology development. *OntoViz*[15] is a *Protégé* plug-in to display an ontology as a 2D graph with capability of showing the names, properties, inheritance, and relations. *TGVizTab*[16] visualizes *Protégé* ontologies in the spring layout where nodes repel one another, whereas the edges (links) attract them. It allows users to navigate gradually making visible parts of the graph. Users may also perform expanding, retracting, hiding, and zooming operations to interact with the graph. *SHriMP*[17] is a domain-independent visualization technique designed to enhance how people browse and explore complex information spaces. *SHriMP* uses a nested graph view and supports concept navigation. One can expand the hierarchy graph by clicking the concept, which provides an intuitive visualization of its relationships. *Jambalaya*[18] is a *Protégé* plug-in based on *SHriMP* with the enhancement of advanced keyword search, allowing users to search the whole ontology or limit the search scope by specifying the type of the searched item. *OntoRama*[19] is an ontology browser for RDF models based on a hyperbolic layout of nodes and arcs. *OntoSphere*[20] is a 3D tool that uses three different ontology views in order to provide an overview or detailed features, such as ancestors, children, and semantic relations. *OAF*[21] is a unified framework and software system for deriving, visualizing, and exploring partial-area taxonomy abstraction networks.

1.4 Conjunctive Navigational Exploration

Searching is the process that helps people locate relevant information from an existing collection. Typically, a user provides a textual query, and the search engine presents a ranked document list in accordance with the computed degree of relevance. However, such searching process is not always effective in supporting navigation and browsing while also locating the targeted results. *Lookup* and *exploratory* are two basic search modes in information retrieval. In the lookup mode, a user knows precisely what to look for, enters search terms, and tries to get the corresponding set of responses. While in the exploratory mode, a user may not be able to easily and effectively formulate a descriptive search term, and must rely on navigational mechanisms such as menus or facets[22] to browse and explore the content. A single search mode may not handle all searching requirements, especially when the volume of search results can be overwhelmingly large and needs to be further structured to allow relevant information to be located. Conjunctive exploratory navigation interfaces CENI and SCENI[23–25] have been developed for exploring consumer health questions with health topics as dynamically search tags complementing keyword-based lookup. The conjunctive exploration mechanism allows users to quickly narrow down to the most relevant results in the most effective way.

2 Methods

WINS follows a Model-View-Controller (MVC) design principle. Figure 2 illustrates the system architecture of WINS. WINS uses MongoDB as the backend database to store non-lattice subgraphs extracted from SNOMED CT. The view mainly contains four parts: a) facet options; b) query widgets; c) result display panel; and d) graph rendering. Controller connects the model and view. Query translator converts user's search requests to mongoDB queries. The query results are then sent to topological sort and position calculation algorithms to obtain formatted data for D3.js to render non-lattice subgraphs.

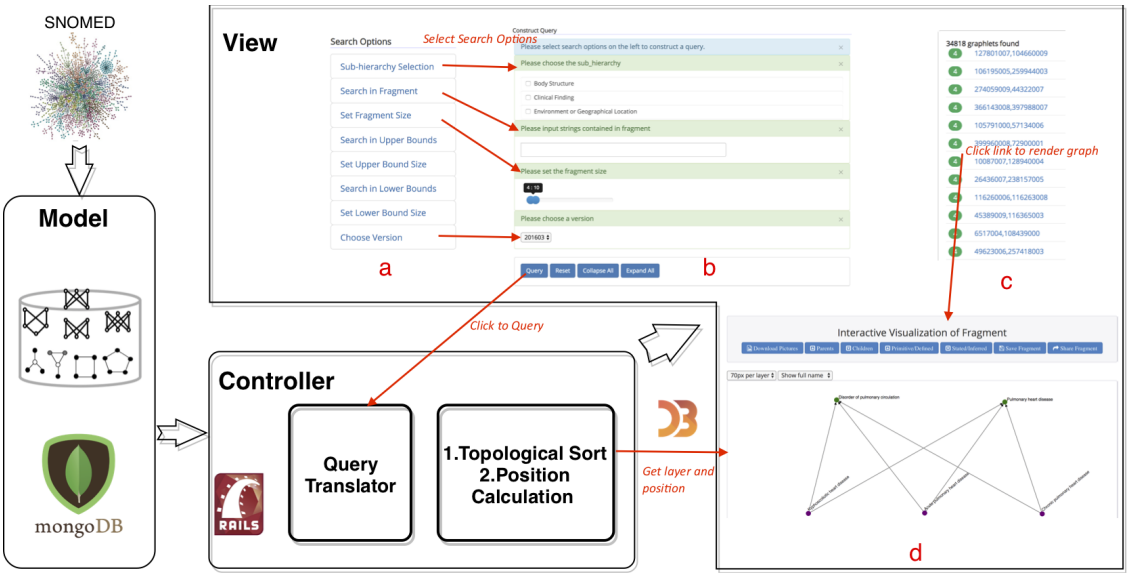

Figure 2: Functional architecture: a. facet options; b. query widgets; c. query results; d. graph rendering.

2.1 Generating Non-Lattice Subgraphs

Non-lattice fragments are graphs consisting of the concepts between any member of the non-lattice pair and any member of their maximal lower bounds[1,3]. However, it is possible that multiple non-lattice pairs have identical maximal lower bounds. In this case, multiple non-lattice fragments will share the same maximal lower bounds, which could involve redundant work in analyzing each of these fragments. To avoid such redundant work, the definition of "non-lattice subgraph" has been introduced[10]. A non-lattice subgraph is determined by a non-lattice pair and its maximal lower bounds. We generate non-lattice subgraphs for WINS by only including the minimal concepts sharing the same maximal lower bounds. Given a non-lattice pair p and its maximal lower bounds $mlb(p)$, a

non-lattice subgraph can be obtained by (1) reversely computing the minimal upper bounds of the maximal lower bounds, denoted by $mub(mlb(p))$, and (2) aggregating all the concepts and edges between (including) any concept in $mub(mlb(p))$ and any of the maximal lower bounds $mlb(p)$[10]. We call $mub(mlb(p))$ and $mlb(p)$ the upper bounds and lower bounds of the non-lattice subgraph, respectively. The size of a non-lattice subgraph is defined as the number of concepts it contains. For instance, given a non-lattice pair (1, 2) and its maximal lower bounds {5, 6} as shown in Figure 3, computing the minimal upper bounds of {5, 6} yields {1, 2, 3}. Then the non-lattice subgraph contains all the concepts and relations between {1, 2, 3} and {5, 6}, that is, concepts {1, 2, 3, 4, 5, 6} and *is-a* relations {(5, 1), (5, 2), (5, 3), (6, 1), (6, 4), (4, 2), (6, 3)}. In this case, we aggregate three non-lattice fragments {1, 2, 4, 5, 6}, {1, 3, 5, 6}, and {2, 3, 4, 5, 6} into one non-lattice subgraph.

2.2 Processing Data Model

We followed our previous work[9] to compute non-lattice pairs and non-lattice subgraphs for each version of SNOMED CT. We store non-lattice subgraphs in MongoDB in the format of document. The advantage of using MongoDB and document format is that we do not need to restrict to specific schemas and can dynamically extend existing data models with customized attributes. To support facet-based conjunctive search and visualization, a non-lattice subgraph document in WINS includes the following information: upper bound concepts and size, lower bound concepts and size, total size of the non-lattice subgraph, all concepts and relations between them, synonyms, hierarchical information, and version. Pre-computed upper and lower closures as a table of triples $(c, {}^{\uparrow}c, {}_{\downarrow}c)$ for each concept c is stored in the database to speed up graph rendering algorithms, where ${}^{\uparrow}c$ denotes concept c's ancestors and ${}_{\downarrow}c$ denotes concept c's descendants.

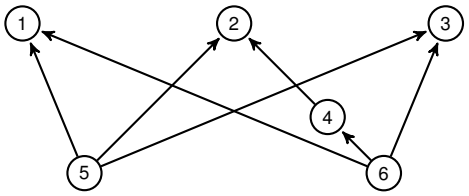

Figure 3: An example of non-lattice subgraph of size 6. Nodes represent concepts and edges represent *is-a* relations.

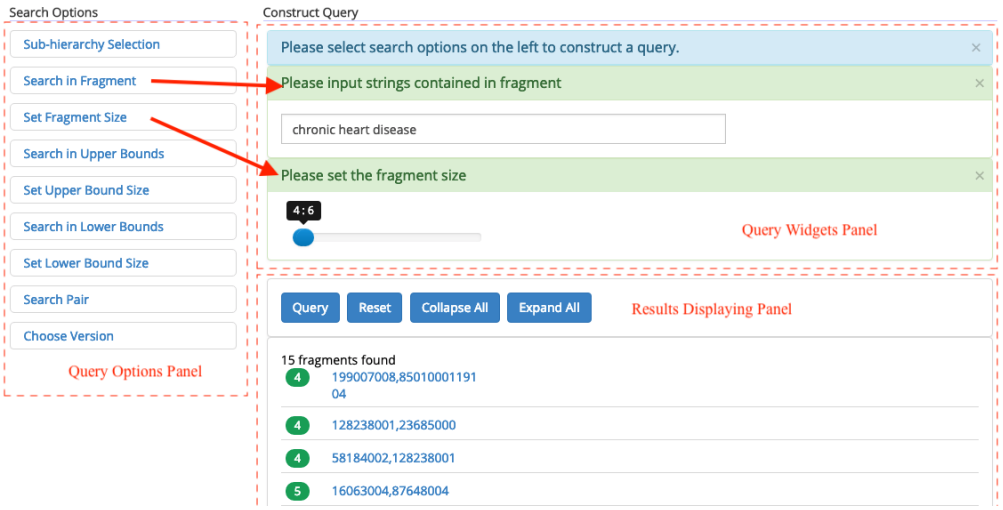

Figure 4: Conjunctive query interface of WINS.

2.3 Conjunctive Query Interface

Benefited from the mechanism developed in CENI[23,24], a novel search method is designed to support fast and precisely locating non-lattice subgraphs in WINS. By leveraging the semantic and structural information in the subgraphs, we create a set of query facets to help narrow down the search scope, such as the subgraph size. Another feature is that words and metadata are extracted and serve as topic tags for each subgraph, which could help improve the relevance of search results. This search method allows the user to navigate to specific non-lattice subgraphs in multiple ways. The WINS search interface supports arbitrary logical conjunction of options and provides the user with multiple paths

to quickly narrow down to the most relevant contents. Figure 4 shows the layout of the conjunctive query interface of WINS. It contains the query options panel and the query widgets construction panel.

Query facets allow users to navigate through different topics of the search results. In WINS, we have 9 facet options (shown in Figure 4): search terms in upper bounds and lower bounds of subgraphs; search terms in concept descriptions of subgraphs; search by the subgraph size, upper bound size, or lower bound size; search in a specific hierarchy; search in a specific SNOMED CT version; and search a pair of concepts. By selecting one option at a time, in an incremental fashion, a user arrives at narrower and narrower content areas that are relevant to all the options selected so far, conjunctively. For example, searching "chronic heart disease" in the concept descriptions of subgraphs obtains 214 non-lattice subgraphs; and further setting the subgraph size as a range of 4 to 6 obtains 15 non-lattice subgraphs.

Each time a facet is selected, the corresponding query widget will pop up in the query construction panel. The user can enter specific search requirements in each of these query widgets. WINS has several types of widgets to make the inputting easier. Text box is used for entering strings; slider bar is used for setting size; drop-down menu is used for selecting versions; and radio button is used for choosing hierarchies. As shown in Figure 4, users can remove one widget by clicking its "X" or remove all selected widgets by clicking the "Reset" button. "Collapse All" and "Expand All" are used when the list of query widgets is too long. By clicking "Query" button, all the query constraints in this area will be translated into MongoDB queries through the controller of WINS.

Algorithm 1: Topological sort for layers.

```
1  Function Topological Sort(C)
       Data: Upper bounds u, nodes N, adj[][]
       Result: layer[]
2      index i = 1
3      layer[u]=i
4      visited[u]=u
5      queue q.push(u)
6      while q not empty do
7          u' = q.pop()
8          i = i + 1
9          foreach n ∈ N do
10             if adj[u'][n] is true and visited[n] is
                 false then
11                 layer[n] = i and visited[n] =
                     true
12                 q.push(n)
13             end
14         end
15     end
16 return layer[]
```

Queries are generated in the backend and executed. Each query widget will be translated into one query. The intersection of all query results are the final result of subgraphs that match the query criteria. Note here, we do not restrict the exact number and the order of words in query. For example, when we search "chronic heart disease," as long as there are three words "chronic", "heart", and "disease" existing in one concept of the subgraph, this subgraph will be included in the query results. To make the query process fast, we implement several optimization strategies. First, we build indexes on all the subgraphs in MongoDB according to each search option and use covered query technique, a query that can be satisfied entirely using an index and does not have to examine any documents. Second, because the description of concepts is stored in the format of an array of words, we use MongoDB's *multikey indexes*, which support efficient queries against array fields. Third, for each query, instead of returning the whole document, we only capture the document ids, which can highly improve the performance of the intersection of multi-queries.

The result displaying panel in Figure 4 shows the result of non-lattice subgraphs related to the constructed query, displayed in the ascending order of subgraph sizes. The number in the green circle denotes the size of the subgraph, and is followed by a hyper-link showing the identifiers of upper bound concepts of the subgraph. When clicking on these identifiers, the subgraph will be rendered in a new page in the browser.

2.4 Interactive Visualization

Visualization in WINS is created using force-layout form of D3.js, one of the most popular dynamic and interactive data visualization libraries. Since ontology and non-lattice subgraphs are directed acyclic graph, hierarchical (top-down) layouts are easy for users to understand. We use Algorithm 1 (*topological sort*) to assign each node in a subgraph a layer number. When rendering the non-lattice subgraph, nodes in the same layer are drawn at the same vertical level. Several heuristic features have been put forth to capture the visual aesthetics of a graph, including minimizing the number of edge crossings, coloring different types of nodes, minimizing overlapping of text, and

maintaining symmetry.

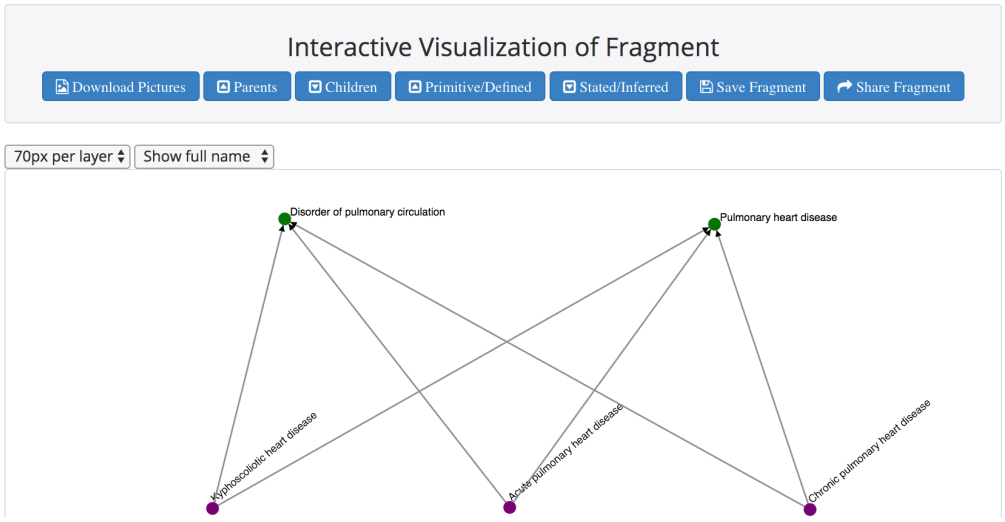

Figure 5: Subgraph rendering page of WINS.

Figure 5 shows the graph rendering page for non-lattice subgraphs. The green nodes represent the upper bounds of the subgraph, the purple ones represent lower bounds, and the gray lines represent *is-a* relations. WINS also provides additional features to make this visualization tool functional and interactive. Users can choose to show the parents of the upper bounds and the children of lower bounds of the subgraph and highlight primitive concepts and stated relations to make them different from defined and inferred ones, which are useful functions for investigating the subgraph. Since the graph is Scalable Vector Graphics (SVG) rendered by D3.js, users can drag the nodes and edges to make the subgraph easy to read, especially for large ones. If a user believes the subgraph is interesting, he or she can save the subgraph for further investigation and also can share the finding to others, by clicking "Save Subgraph" and "Share Subgraph" button respectively. Besides, user can download the graph in various formats including SVG, PNG and TIKZ. Subgraph editing function is also available in WINS.

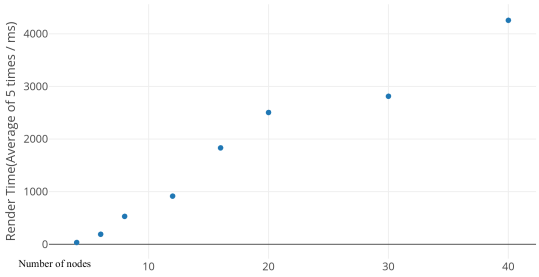

Figure 6: Non-lattice subgraph rendering time.

Tools	D	S	Z	C	E	NL
WINS	✓	✓	✓	✓	✓	✓
SNOMED Browser	✗	✓	✗	✗	✗	✗
SHriMP	✗	✓	✓	✗	✗	✗
OAF	✗	✓	✗	✗	✓	✗
Protégé	✗	✓	✗	✗	✓	✗
Web-Protégé	✗	✓	✗	✓	✓	✗
Jambalaya	✗	✓	✓	✗	✗	✗
OntoSphere	✗	✗	✓	✗	✗	✗

Figure 7: Comparison of the selected ontology visualization tools (D: Drag, S: search, Z: zoom, C: collaborative, E: edit, NL: non-lattice).

3 Results

We used the distribution files of SNOMED CT (US edition) to generate non-lattice subgraphs. From the March 2012 version to the September 2018 version, we have generated 14 versions of non-lattice subgraphs and stored them into the database, along with the corresponding concepts, relations, and term dictionaries.

3.1 Performance Evaluation of the Layout Algorithm

The time complexity of the layout algorithm needs to be low to maintain efficient interaction with users. We randomly chose 40 subgraphs (5 subgraphs for each size of 4, 6, 8, 12, 16, 20, 30, 40) and took the average rendering time to evaluate the performance. Figure 6 shows the results. Even for size of 40, it only takes about 4ms, which indicates a reasonable rendering time.

3.2 Feature Comparison with Other Ontology Visualization Tools

We compared WINS with several other ontology visualization tools, including SNOMED CT Browser[4], SHriMP[17], OAF[21], Protégé[13], Web-Protégé[14], Jambalaya[18], and OntoSphere[20]. The comparison mainly focused on features and functions for non-lattice-based OQA, including drag, search, zoom, collaborative, editing, and non-lattice display. Table 7 lists those where we have found software prototypes. A check mark denotes the tool has the corresponding feature, while cross mark denotes it does not have the feature. The functions of drag, zoom, and edit can improve the ease of analysis of non-lattice subgraphs in WINS.

3.3 OQA Work Supported by WINS

Non-lattice subgraphs are generated and imported into WINS. With WINS's conjunctive explore query interface and graph rendering technique, ontology experts can easily find and view specific subgraphs, which dramatically reduces the effort spent on quality assurance work compared to reading text-based descriptions. Two examples of OQA works supported by visualization of non-lattice subgraphs are given in the following of this subsection.

Visualizing relation reversals in ontological evolution. The visualization of SNOMED CT's hierarchical non-lattice fragments is important not only for showing what has been asserted in relations, but also for demonstrating what is NOT asserted. WINS provides a platform not only to perform exhaustive structural analysis of an individual SNOMED CT version, but also to systematically track structural changes between versions. We demonstrated the using of visualization to detect flip-flops in the evolution of an ontological system, which are direct reversals of relations from one version to another in our previous work[26]. When "A *is-a* B" in one version is reversed to "B *is-a* A" in another version, which is an important structural change and could be a potential place that errors occur.

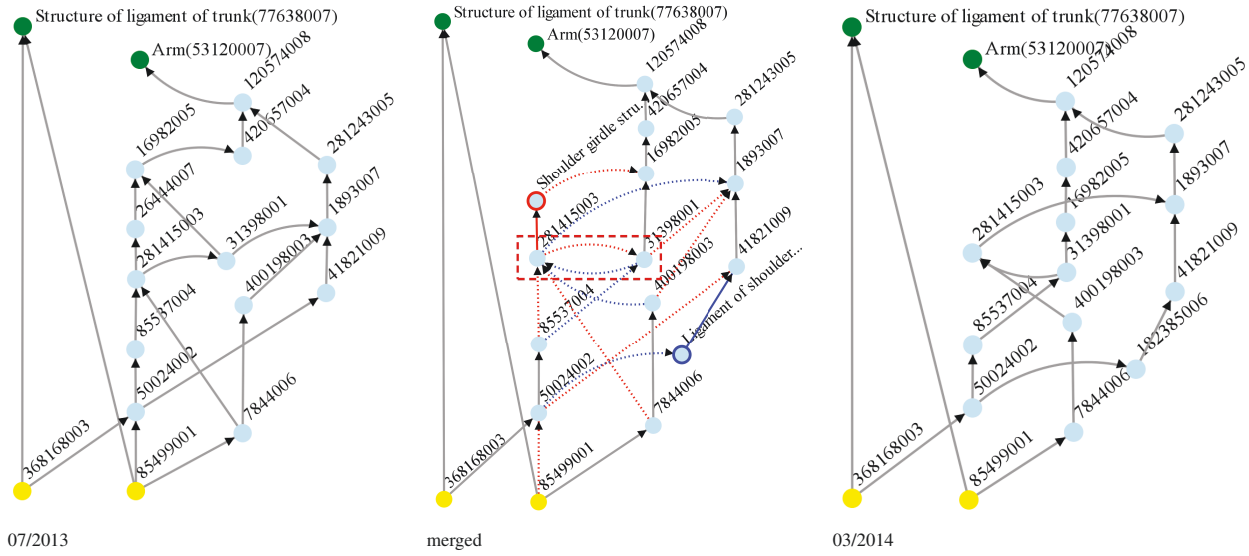

Figure 8: Non-lattice subgraphs from two SNOMED CT versions and their merged graph[27]. Left: a non-lattice subgraph of 7/2013 version of SNOMED CT. Right: a non-lattice subgraph of 03/2014 version of SNOMED CT. Middle: merged graph showing the changes.

Figure 8 demonstrates an example non-lattice fragment around concepts "Arm (53120007)" and "Structure of ligament of trunk (77638007)" from two SNOMED CT versions, 07/2013 and 03/2014. A merged graph[27] of the two subgraphs is shown in the middle. Green nodes represent non-lattice pairs and yellow nodes represent maximal lower bounds of the non-lattice pairs. All nodes in-between non-lattice pairs and their maximal lower bounds are in light blue. Gray lines represent *is-a* relations. In the merged graph, additional graph notations are introduced to represent changes with red denoting deletion and blue denoting insertion. Nodes with solid red borders and solid red edges represent deletion, which means these appear in old fragment but not new. Similarly, nodes with solid blue borders and solid blue edges represent addition, which means these appear in new fragment but not old. Nodes with dashed borders and dashed edges represent insertion and deletion appear in respective fragments in the two versions. With the merged graph and these dedicated graph notations, one can easily find a loop (inside the red dotted rectangle) in the graph involving one reversal pair of of "Joint structure of shoulder girdle (281415003)" and "Joint structure of shoulder region (31398001)". The *is-a* relation changed reversely between the two versions.

Detecting missing hierarchical relations and concepts using a structural-lexical method. Visualization of non-lattice subgraphs can transfer the text descriptions of concepts and relations into a more straightforward representation format, which can facilitate interpretation and reduce effort spent in analyzing and evaluation. In our previous work[10], four lexical patterns were identified among the extracted non-lattice subgraphs: containment, intersection, union, and union-intersection. Non-lattice subgraphs exhibiting such lexical patterns are often indicative of missing hierarchical relations or concepts. Visualization of samples of the potential errors detected, as well as the proposed remediations in WINS, are provided to experts for review and validation. Take the containment pattern as an example, which is generally an indicative of a missing *is-a* relation between concepts in the upper bounds or lower bounds. The non-lattice subgraph shown in Figure 9A exhibits a containment pattern between the two concepts in the lower bounds, that is, {*duodenal, ulcer, with, perforation, and, obstruction*} is contained in {*chronic, duodenal, ulcer, with, perforation, and, obstruction*}. Here, there is a missing hierarchical relation between these two concepts, because the added notion of *chronic* makes *Chronic duodenal ulcer with perforation AND obstruction* more specific. Figure 9B shows the suggested correction for the non-lattice subgraph, that is, adding the relation *Chronic duodenal ulcer with perforation AND obstruction* ***is-a*** *Duodenal ulcer with perforation AND obstruction*. This example showed that the visualization of subgraphs can support OQA works, especially for experts to review and analyze.

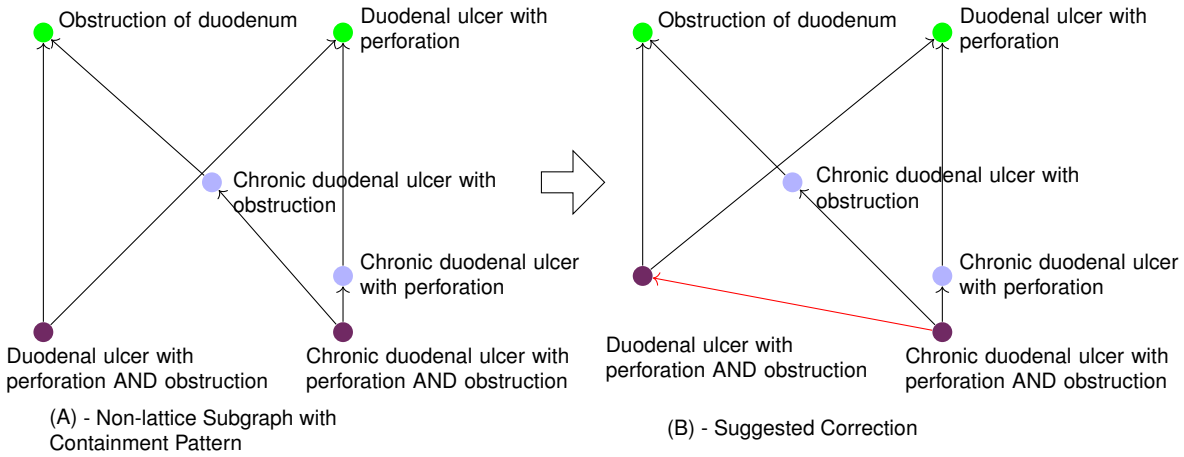

Figure 9: A non-lattice subgraph in the September 2015 US edition of SNOMED CT (left) and its suggested remediation (right).

4 Discussion

Interactive Visualization. Although WINS provides a practical and efficient method for users to access and visualize non-lattice subgraphs, there are limitations as well. It highly relies on domain experts to manually check each subgraph and provide remediation suggestions. One possible solution is to use Formal Concept Analysis[1] to automatically

suggest missing concepts and relations for each subgraph. Other lexical or structural auditing methods can also be good supplements to enhance WINS to support automatic or semi-automatic ontology quality assurance work.

Generalizability. WINS was originally designed only for SNOMED CT, but its general and flexible system architecture allows it to be able to serve as a framework for similar systems that can visualize non-lattice subgraphs for other ontologies such as Gene Ontology.

Collaboration. WINS provides a community function for users to share the assessment of their findings, so change recommendations can be accessed by SNOMED CT editorial committee directly. In the "share" page, users can post a subgraph and the correction suggestions. Other users can join the discussion and make their comments. Voting function is also included. Basically, it functions like a common forum.

5 Conclusions

This paper presents WINS, an interactive platform for visualizing the structures of ontologies to support non-lattice based ontology quality assurance and shared decision making. The purpose of WINS is to provide a browser for users to access, retrieve, and visualize non-lattice subgraphs in identifying the most promising areas for quality assurance. WINS is designed to serve as a community portal for SNOMED CT developers and users to perform collaborative graph-sensemaking and change recommendation. Our preliminary results suggest that WINS has fulfilled these design objectives.

Acknowledgment

This work was supported by the National Science Foundation (NSF) through grant IIS-1931134 and the National Institutes of Health (NIH) National Cancer Institute through grant R21CA231904. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NSF or NIH.

References

1. Guo-Qiang Zhang and Olivier Bodenreider. Large-scale, exhaustive lattice-based structural auditing of SNOMED CT. *AMIA Annual Symposium Proceedings*, volume 2010, 922–926.

2. Pierre Zweigenbaum, Bruno Bachimont, Jacques Bouaud, Jean Charlet, and J-F Boisvieux. Issues in the structuring and acquisition of an ontology for medical language understanding. *Methods of Information in Medicine*, 34(01/02):15–24, 1995.

3. Guo-Qiang Zhang, Wei Zhu, Mengmeng Sun, Shiqiang Tao, Olivier Bodenreider, and Licong Cui. MAPLE: a mapreduce pipeline for lattice-based evaluation and its application to SNOMED CT. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 754–759. IEEE, 2014.

4. Jeremy Rogers and Olivier Bodenreider. SNOMED CT: Browsing the browsers. In *KR-MED*, pages 30–36, 2008.

5. NLM SNOMED CT Browser. `https://uts.nlm.nih.gov/`. Accessed: 2019-01.

6. Michael Burch and Steffen Lohmann. Visualizing the evolution of ontologies: A dynamic graph perspective. In *VOILA@ ISWC*, page 69, 2015.

7. Snomed ct. `http://www.snomed.org/`. Accessed: 2019-01.

8. Health information technology for economic and clinical health. `http://www.healthit.gov/sites/default/files/hitech_act_excerpt_from_arra_with_index.pdf`. Accessed: 2019-01.

9. Guo-Qiang Zhang, Guangming Xing, and Licong Cui. An efficient, large-scale, non-lattice-detection algorithm for exhaustive structural auditing of biomedical ontologies. *Journal of Biomedical Informatics*, 80:106–119, 2018.

10. Licong Cui, Wei Zhu, Shiqiang Tao, James Case, Olivier Bodenreider, and Guo-Qiang Zhang. Mining non-lattice subgraphs for detecting missing hierarchical relations and concepts in SNOMED CT. *Journal of the American Medical Informatics Association*, 24(4):788–798, 2017.

11. Sergey Mikhailov, Mikhail Petrov, and Birger Lantow. Ontology visualization: A systematic literature analysis. In *BIR Workshops*, 2016.

12. Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou. Ontology visualization methods:a survey. *ACM Computing Surveys (CSUR)*, 39(4):10, 2007.

13. Natalya Fridman Noy, Monica Crubézy, Ray W Fergerson, Holger Knublauch, Samson W Tu, Jennifer Vendetti, and Mark A Musen. Protégé-2000: an open-source ontology-development and knowledge-acquisition environment. In *AMIA Annual Symposium Proceedings*, volume 2003, pages 953–953.

14. Steffen Lohmann, Vincent Link, Eduard Marbach, and Stefan Negru. Webvowl: Web-based visualization of ontologies. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 154–158. Springer, 2014.

15. Michael Sintek. Ontoviz tab: Visualizing protégé ontologies, 2003, 2013.

16. Harith Alani. Tgviztab: an ontology visualisation extension for protégé. 2003.

17. Margaret-Anne Storey, Casey Best, Jeff Michaud, Derek Rayside, Marin Litoiu, and Mark Musen. Shrimp views: an interactive environment for information visualization and navigation. In *CHI'02 Extended Abstracts on Human Factors in Computing Systems*, pages 520–521. ACM, 2002.

18. Margaret-Anne Storey, Mark Musen, John Silva, Casey Best, Neil Ernst, Ray Fergerson, and Natasha Noy. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in protégé. In *Workshop on interactive tools for knowledge capture*, volume 73, 2001.

19. Peter Eklund, Nataliya Roberts, and Steve Green. Ontorama: Browsing rdf ontologies using a hyperbolic-style browser. In *Cyber Worlds, 2002. Proceedings. First International Symposium on*, pages 405–411. IEEE, 2002.

20. Alessio Bosca, Dario Bonino, and Paolo Pellegrino. Ontosphere: more than a 3d ontology visualization tool. In *Swap*. Citeseer, 2005.

21. Christopher Ochs, James Geller, Yehoshua Perl, and Mark Musen. A unified software framework for deriving, visualizing, and exploring abstraction networks for ontologies. *J. Biomedical Informatics*, 62:90–105, 2016.

22. Guo-Qiang Zhang, Shiqiang Tao, Ningzhou Zeng, and Licong Cui. Ontologies as nested facet systems for human-data interaction. *Semantic Web*, In Press.

23. Licong Cui, Shiqiang Tao, and Guo-Qiang Zhang. A semantic-based approach for exploring consumer health questions using umls. In *AMIA Annual Symposium Proceedings*, volume 2014, page 432. American Medical Informatics Association, 2014.

24. Licong Cui, Rong Xu, Zhihui Luo, Susan Wentz, Kyle Scarberry, and Guo-Qiang Zhang. Multi-topic assignment for exploratory navigation of consumer health information in netwellness using formal concept analysis. *BMC medical informatics and decision making*, 14(1):63, 2014.

25. Licong Cui, Rebecca Carter, and Guo-Qiang Zhang. Evaluation of a novel conjunctive exploratory navigation interface for consumer health information: a crowdsourced comparative study. *Journal of Medical Internet Research*, 16(2), 2014.

26. Shiqiang Tao, Licong Cui, Wei Zhu, Mengmeng Sun, Olivier Bodenreider, and Guo-Qiang Zhang. Mining relation reversals in the evolution of snomed ct using mapreduce. *AMIA Summits on Translational Science Proceedings*, 2015:46, 2015.

27. Licong Cui, Shiqiang Tao, and Guo-Qiang Zhang. Biomedical ontology quality assurance using a big data approach. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(4):41, 2016.