FEDRR: Fast, Exhaustive Detection of Redundant Hierarchical Relations in Large Biomedical Ontologies

Guangming Xing[1], Licong Cui[2], and Guo-Qiang Zhang[2]*

[1]Department of Computer Science
Western Kentucky University, Bowling Green, KY 42101, USA
[2]Institute of Biomedical Informatics
University of Kentucky, Lexington, KY 40506, USA
guangming.xing@wku.edu
{licong.cui,gqatcase}@gmail.com

Abstract. Redundant hierarchical relations refer to such patterns as two paths from one concept to another, one with length one (direct) and the other with length greater than one (indirect). This paper introduces a novel and scalable approach, called FEDRR – Fast, Exhaustive Detection of Redundant Relations – for quality assurance work during ontological evolution. FEDRR combines the algorithm ideas of Dynamic Programming with Topological Sort, for exhaustive mining of all redundant hierarchical relations in ontological hierarchies, in $O(c \cdot |V| + |E|)$ time, where $|V|$ is the number of concepts, $|E|$ is the number of the relations, and c is a constant in practice. Using FEDRR, we performed exhaustive search of all redundant is-a relations in two of the largest ontological systems in biomedicine: SNOMED CT and Gene Ontology (GO). 235 and 1609 redundant is-a relations were found in the 2015-03-01 version of SNOMED CT and 2015-05-01 version of GO, respectively. Each redundant relation represents a possibly unintended defect that needs to be corrected in the ontology quality assurance process. FEDRR provides a generally applicable, effective tool for systematic detecting redundant relations in large ontological systems for quality improvement.

Keywords: Redundant relations, SNOMED CT, Gene Ontology, Dynamic Programming

1 Introduction

Ontologies are shared conceptualizations of a domain represented in a formal language. They represent not only the concepts (nodes) but the relationships (edges) between the concepts. Ontologies have become a critical knowledge source in informatics and data intensive applications, such as information retrieval [1], data integration [2], data management [3], and decision support [4].

This paper focuses on a particular type of ontological structural defect: redundant relations. Redundant hierarchical relations refer to such patterns as two paths from concept X to concept Y, one with length one (direct) and the other with length greater than one (indirect). For hierarchical relations such as subsumption (is-a), relations implied by transitivity should not be explicitly stated. For example, in Gene Ontology (GO 2015-05-01 version) we have (see Table 1):

* Corresponding author.

	GO Id	Relation		GO Id
A	GO:0046879	is-a	B	GO:0009914
B	GO:0009914	is-a	C	GO:0010817
C	GO:0010817	is-a	D	GO:0065008
D	GO:0065008	is-a	E	GO:0065007
E	GO:0065007	is-a	F	GO:0008150

Table 1: A="hormone secretion;" B="hormone transport;" C= "regulation of hormone levels;" D="regulation of biological quality;" E="biological regulation;" F= "biological process."

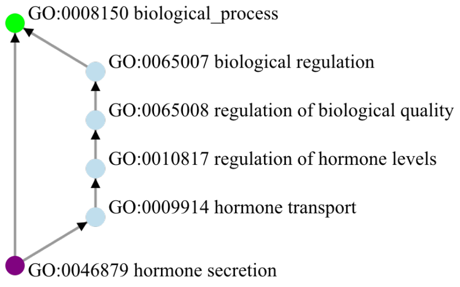

GO:0008150 biological_process
GO:0065007 biological regulation
GO:0065008 regulation of biological quality
GO:0010817 regulation of hormone levels
GO:0009914 hormone transport
GO:0046879 hormone secretion

Fig. 1: Graphical rendering of Table 1 and a direct edge between A and F, where directed edges represent "is-a" relation.

However, "A (GO:0046879) is-a F (GO:0008150)" is directly asserted as well (Fig. 1). This represents redundant relations to be studied in this paper: two paths exist between A and F: one directly between A and F, and the other indirectly through B, C, D, and E as intermediate concept nodes.

The principle of parsimony in ontological modeling refers to the omission of relations implied by the transitive property of a relationship, such as "is-a" relations in GO. By violating this principle, redundant relations may increase maintenance burden for ontology curators. It can also cause and inaccurate methods and algorithms based on this general principle. For example, semantic distance between concepts is a widely used technique [5]. Ontological mapping and alignment methods rely on the ordered structure of the hierarchical relation [6], with notions of neighborhood and proximity serving as their foundation. The presence of redundant relations induce a short-circuit: two concepts with a larger semantic distance may result in a smaller distance by mistake; and concepts not within a neighborhood may be counted as such.

Using brute force, exhaustive detection of redundant relations can be computationally expensive for large ontologies. For example, SNOMED CT (2015-03-01 version) contains over 300,000 active concepts. A naive approach would be to find the longest paths between the end nodes of each of the over 500,000 edges (relations). *Assuming each edge takes 100ms, processing a single version of SNOMED CT would take 14 hours.* Finding all paths between all possible pairs among the 300k nodes would take over 10,000 days if each pair takes 10ms.

This paper introduces a novel and scalable approach, called FEDRR, Fast, Exhaustive Detection of Redundant Relations, for quality assurance work during ontological evolution. In contrast to the 14 hours naive approach required for each SNOMED CT version, *FEDRR needed <20 seconds (Section 4).*

Using FEDRR, we performed exhaustive search of all redundant is-a relations in two of the largest ontological systems in biomedicine: SNOMED CT and GO. 235 and 1609 redundant is-a relations were found in the most recent versions of SNOMED CT and GO, respectively. Each redundant relation represents a possibly unintended defect that needs to be corrected in the ontology quality assurance process. We further performed longitudinal analyses using FEDRR on 5 recent versions of SNOMED CT and 10 versions of GO.

2 Background

SNOMED CT. SNOMED CT is the world's largest clinical terminology [7, 8]. It provides broad coverage of clinical medicine, including findings, diseases, and procedures for use in electronic medical records. 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" including Procedure, Substance, Body structure, Specimen, Clinical finding, and Organism. 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.

Gene Ontology. The Gene Ontology [9] is a collection of three ontologies to describe attributes of gene products in three non-overlapping domains of molecular biology: Cellular Component, the parts of a cell or its extracellular environment; Molecular Function, the elemental activities of a gene product at the molecular level, such as binding or catalysis; and Biological Process, operations or sets of molecular events with a defined beginning and end, pertinent to the functioning of integrated living units (cells, tissues, organs, and organisms). Within each ontology, terms have free text definitions and unique identifiers. GO terms can be related to each other by is-a and part-of relationships, forming a directed acyclic graph. The GO vocabulary is designed to be species-agnostic, and is intended to capture multiple organisms.

Ontology Quality Assurance. Large, comprehensive terminological systems such as SNOMED CT and GO continue to evolve over time [12–19]. Ontology Quality Assurance (OQA) is an indispensable part of the ontological engineering lifecycle [10, 11]. OQA attempts to assess and improve the overall quality of ontologies in aspects such as the consistency of the ontological structure with respect to the explicit and implicit knowledge they capture; the coverage of the ontology in terms of classes and properties needed to support specific applications; and the non-redundancy of classes and properties.

The basic premise of OQA is a mixed closed-world assumption (CWA) and open-world assumption (OWA). In a formal system of logic used for knowledge representation, such as ontological systems, CWA refers to the assumption that a relationship holds true between two concepts is also explicitly asserted to be true, unless they are implied by logical properties such as transitivity. It dictates that, in reverse, a relationship between two concepts that is not asserted explicitly, must be false. OWA, on the other hand, refers to the assumption that lack of knowledge does not imply falsity.

In the context of OQA, OWA refers to the evolving state of knowledge in a domain, in the sense that new concepts may be included in an ontological system in a continuous fashion. The lack of a concept in an ontological system does not imply that such a concept does not exist. CWA, on the other hand, implies that, among existing concepts in an ontological system, the lack of an explicit relationship of a known relation-type between two concepts means that such a relationship does not exist between the two concepts.

The *principle of parsimony* in ontological modeling is a direct consequence of CWA. It refers to the fact that relations implied by the transitive property of a relationship, such as the example given in Fig. 1, must not be explicitly stated. By violating this principle, redundant relations can cause methods and algorithms based on this general principle inaccurate. Detecting redundant relations is an important task for OQA, which is the focus of this paper.

3 Methods

The general mathematical abstraction of an ontological structure is a graph-theoretic one: nodes correspond to concepts, and edges correspond to relations (between nodes). For hierarchical relations in ontological systems such as "is-a," which obeys the *transitivity property* that

if A is-a B and B is-a C, then A is-a C,
one can model the structure of an ontological system as a directed acyclic graph (DAG, as shown in part in Fig. 1).

Definition 1. *Suppose $G = (V, E)$ is a directed acyclic graph with V a set of nodes, and E a set of edges between the nodes. A redundant relation in G is a pair of nodes (s, t) such that $(s, t) \in E$, and there is an indirect path (i.e., length more than 1) from s to t.*

The closely related known algorithm for computing redundant relations in the literature is all-pair longest path [20]. Although fixed source longest path can be solved in time-complexity $O(|V| + |E|)$ in a DAG [20], all-pair longest path requires iteration over V, resulting in an $O(|V| \cdot |E| + |V|^2)$ time-complexity algorithm. For large ontological systems such as SNOMED CT, such a running time amounts to an intractable amount of processing time (requiring 10,000 days if all-pair paths were to be computed).

FEDRR solves this problem in time-complexity $O(c \cdot |V| + |E|)$, where c is the average number of descendants of a node. For the latest version of SNOMED CT, we have $c = 17.12$ (see Time Complexity Analysis). For a single version of SNOMED CT, the actual processing time is less than *20 seconds.*

There are two key algorithmic ideas behind FEDRR. One is avoidance of repeated computations by remembering the set of directly reachable nodes as well as the set of indirectly reachable nodes, for each node. The second is to completely skip node pairs that are not connected by a directed path. These ideas are reflected in FEDRR using a novel combination of dynamic programming with topological sort. The sparsity of most ontological structures, viewed as a DAG, is a particularly suitable property for the second idea to take advantage of.

For a node u in a DAG $G = (V, E)$, we introduce two sets, D_u and I_u, where

- $D_u = \{v \mid (v, u) \in E\}$, called the ***D*-set**, consists of the direct descendants (i.e. children) of u; and
- I_u, called the ***I*-set** of u, is the set of all indirect descendants of u.

The design of our algorithm is based on the following observation.

Lemma 1. *For each node $v \in D_u \cap I_u$, (v, u) is redundant.*

Our algorithm amounts to the computation of (D_u, I_u) for each node u. To utilize the idea of dynamic programming, we update (D_u, I_u) for each node u according to the order by topological sort. The basic update scheme is illustrated in the following diagram:

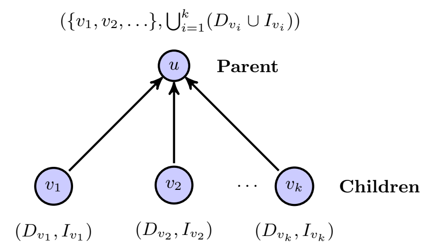


Fig. 2: Basic mechanism for updating the D-set and the I-set of a node.

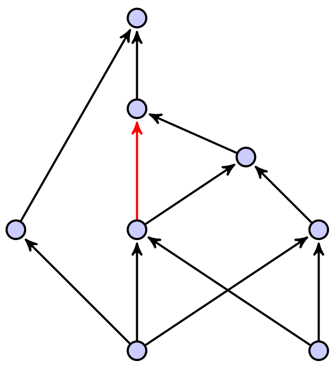
Suppose we have obtained (D_{v_i}, I_{v_i}) for each $i = 1, \ldots, k$, where $\{v_1, v_2, \ldots\} = \{v \mid (v, u) \in E\}$. Then we set $D_u = \{v_1, v_2, \ldots\}$ and $I_u = \bigcup_{i=1}^{k}(D_i \cup I_i)$. The pseudo-code of FEDRR appears in Algorithm 1.

Algorithm 1 FEDRR: Dynamic programming using topological sort to compute the D-set and I-set of each node

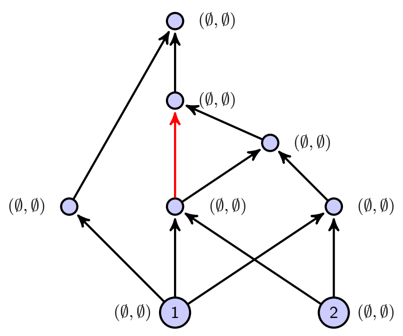
```
1:  Input: $G(V)$
2:  $q := new\ Queue()$
3:  for all $v \in V$ do
4:      $I[v] := \emptyset$
5:      $D[v] := \emptyset$
6:      if no incoming edge for $v$ then
7:          $q.enqueue(v)$
8:      end if
9:  end for
10: while q not empty do
11:     $s := q.dequeue()$
12:     for all $t \in s.to$ do
13:         $I[t] := I[t] \cup I[s] \cup D[s]$
14:         $D[t] := D[t] \cup \{s\}$
15:         mark edge $(s, t)$
16:         if no unmarked incoming edge for $t$ then
17:             $q.enqueue(t)$
18:         end if
19:     end for
20: end while
```

FEDRR starts by initializing an empty queue to hold the nodes that will be sorted (line 2). Then nodes with no incoming edges are put to the queue, with the D-set and I-set initialized as empty (lines 3 - 9). In the next phase (lines 10 - 20), the nodes are dequeued one at a time, with the I-sets and D-sets (for t) updated according to the mechanism described in Fig. 2.

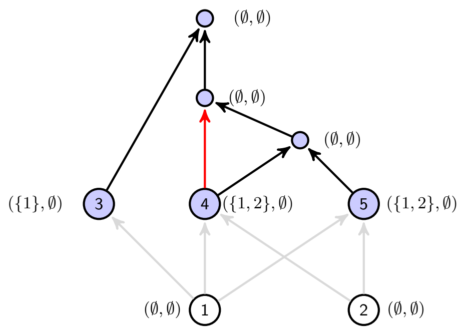
We illustrate the steps of Algorithm 1 using an example. The input DAG is given below, and there is a redundant edge (colored in red) that FEDRR is supposed to detect.



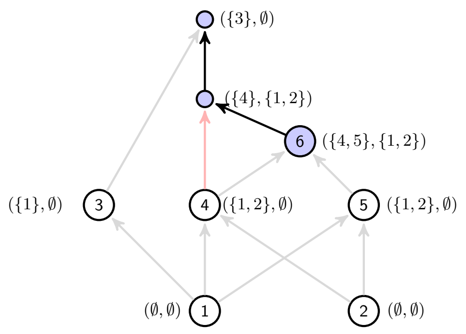
The algorithm starts with setting initial values for the D-set and the I-set and enqueuing those node with no incoming edges, as shown on the top of Fig. 3 on the right.



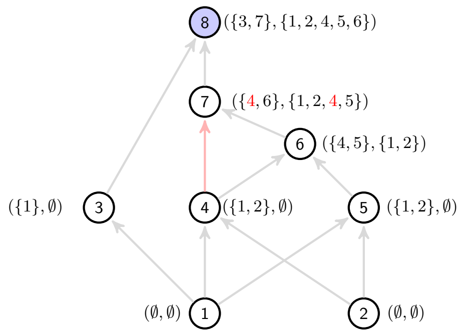
After lines 2 - 9, nodes $1, 2$ are enqueued.



Nodes $1, 2$ dequeued, $D-$set and $I-$set updated on $3, 4, 5$. Nodes $3, 4, 5$ enqueued.



After nodes 3, 4, 5 dequeued, D-set and I-set updated on nodes 6, 7, 8 (7, 8 not enqueued yet, thus not numbered). Node 6 enqueued.



Node 6 dequeued, D-set and I-set updated on node 7. Node 7 enqueued. Node 7 dequeued, D-set and I-set updated on node 8. Node 8 enqueued.

Fig. 3: Illustration of Algorithm 1.

For this sample DAG, the result is shown in Fig. 4.

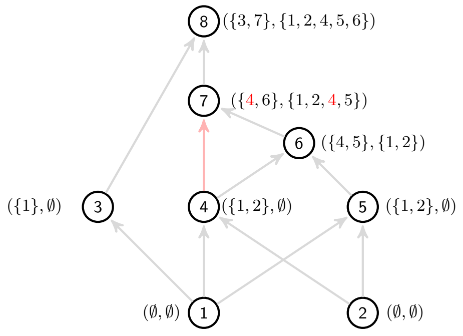


Fig. 4: Node 8 dequeued, queue is empty.

Correctness. The correctness of the algorithm can be proved using mathematical induction by showing $I[v_i] = I_{v_i}$ and $D[v_i] = D_{v_i}$ after node v_i is dequeued (line 11) for $i = 1 \ldots |V|$.

Proof. $i = 1$. The first dequeued node must be a node with no incoming edges. This means $I_{v_1} = \emptyset$ and $D_{v_1} = \emptyset$. As both $I[v_1] = \emptyset$ and $D[v_1] = \emptyset$ from lines 4 and 5, we have $I[v_1] = I_{v_1}$ and $D[v_1] = D_{v_1}$.

Suppose $I[v_i] = I_{v_i}$ and $D[v_i] = D_{v_i}$ is true for $i = 1 \ldots k-1$. For $i = k$, then we have $D[v_k] = \{v \mid (v, v_k) \in E\}$ and $I[v_k] = \bigcup_j (D[v_{k_j}] \cup I[v_{k_j}])$, where $v_{k_j} \in \{v \mid (v, v_k) \in E\}$. Based on the definition of D_v, we have $D_{v_k} = \{v \mid (v, v_k) \in E\} = D[v_k]$. From the induction hypothesis, we have $I[v_i] = I_{v_i}$ and $D[v_i] = D_{v_i}$ for $i = 1 \ldots k-1$. This means $I[v_k] = \bigcup_j (D[v_{k_j}] \cup I[v_{k_j}]) = \bigcup_j (D_{v_{k_j}} \cup I_{v_{k_j}}) = I_{v_k}$.

Time Complexity Analysis. The topological sorting itself takes $O(|V| + |E|)$ time [21]. With the computation of D-set and I-set, the total time is $O(\sum_{(u,v)\in E}(|D_v| + |I_v|) + |V| + |E|)$. When $|E| = O(|V|)$ (which is the case for both SNOMED CT and GO), the running time is $O(\sum_v(|D_v| + |I_v|) + |V| + |E|)$. If we let $c = \frac{\sum_v(|D_v|+|I_v|)}{|V|}$, then the running time is in $O(c \cdot |V| + |E|)$. Based on the definition of D_v and I_v, $\sum_v(|D_v| + |I_v|)$ is the size of transitive closure pairs shown in Tables 2 and 4. Even though the worst-case running time is $O(|V|^2)$(when $c = |V|$), c is a relatively small constant for ontological systems in practice. This is validated by our experimental results shown in Tables 2 and 4. For the latest version of SNOMED CT, $c = \frac{5{,}408{,}010}{315{,}904} = 17.12$, and for the latest version of GO, $c = \frac{557{,}550}{42{,}979} = 12.97$.

4 Results

4.1 Experimental Environment

To detect redundant is-a relations from SNOMED CT and Gene Ontology, we ran the FEDRR method on a MacBook Pro running the Mac OS X Yosemite with 16 GB RAM and Intel Core i7 processor. FEDRR was implemented in Java programming language based on JDK7.

4.2 Redundant is-a relations in SNOMED CT

We ran the FEDRR method on 5 versions of SNOMED CT (US edition) from 2013 to 2015 dated on 2013-03-01, 2013-09-01, 2014-03-01, 2014-09-01, and 2015-03-01. Table 2 summarizes the result of each version including numbers of concepts, is-a relations, and transitive closure pairs (TC), and number of redundant is-a relations (RR); percentage of redundant is-a relations (RR%) among transitive closure pairs; and computing time in milliseconds to detect redundant is-a relations. For example, for the 2015-03-01 version, there were 315,904 concepts, 467,799 is-a relations, 5,408,010 transitive closure pairs, and 235 redundant is-a relations; the percentage of the redundant is-a relations among the transitive closure pairs is 0.00435%; and it took about 15 seconds to complete. For each version, it only took a few seconds to identify all the redundant is-a relations, indicating the efficiency of FEDRR.

Table 2: Summary of the results for 5 versions of SNOMED CT. TC: number of transitive closure pairs, RR: number of redundant is-a relations, T(ms): time taken in milliseconds.

Version	**# Concepts**	**# is-a Relations**	**TC**	**RR**	**RR%**	**T(ms)**
2013-03-01	299,198	444,565	5,165,131	203	0.00393	10,874
2013-09-01	300,485	447,442	5,226,630	240	0.00459	10,472
2014-03-01	300,409	446,603	5,188,221	277	0.00534	10,335
2014-09-01	302,902	449,564	5,222,506	305	0.00584	10,074
2015-03-01	315,904	467,799	5,408,010	235	0.00435	15,264

Table 3 shows the numbers of redundant is-a relations in 5 versions of SNOMED CT with respect to the length of the indirect path. For each version, $\mathbf{l_i} (i = 2, 3, 4)$ is the number of redundant is-a relations in length of i regarding to the indirect path. For example, in the version of 2015-03-01, there were 224 redundant is-a relations in length of 2, 10 in length of 3, and 1 in length of 4. In general, most redundant is-a relations were in length of 2, and no redundant is-a relations exceeding length of 4 was identified.

Table 3: Numbers of redundant is-a relations in 5 versions of SNOMED CT regarding to the length of the indirect path. $\mathbf{l_i}$ represents the number of redundant is-a relations in length of i regarding to the indirect path.

Version	$\mathbf{l_2}$	$\mathbf{l_3}$	$\mathbf{l_4}$	**Total**
2013-03-01	199	4	0	203
2013-09-01	233	7	0	240
2014-03-01	264	11	2	277
2014-09-01	291	13	1	305
2015-03-01	224	10	1	235

4.3 Redundant is-a relations in Gene Ontology

We ran the FEDRR method to detect redundant is-a relations in 10 versions of Gene Ontology from 2014-08-01 to 2015-05-01 updated monthly. Table 4 summarizes the basic results of each version. For instance, for the 2015-05-01 version, there were 42,979 concepts, 71,954 is-a relations, 557,550 transitive closure pairs, and 1,609 redundant is-a relations; the percentage of the redundant is-a relations among the transitive closure pairs is 0.2886%; and it took 1,538 milliseconds to complete. As the number of concepts and is-a relations were increasing, the number and percentage of redundant is-a relations (RR) were monotonically increasing every month and increased more than twice from the 2014-08-01 version (497; 0.0961%) to the 2015-05-01 version (1,609; 0.2886%). For each version, it only took a couple of seconds to identify all the redundant is-a relations, indicating the efficiency of FEDRR.

Table 4: Summary of the results for 10 versions of Gene Ontology. TC: number of transitive closure pairs, RR: number of redundant is-a relations, RR%: percentage of redundant is-a relations among transitive closure pairs, T(ms): time taken in milliseconds.

Version	**# Concepts**	**# is-a Relations**	**TC**	**RR**	**RR%**	**T(ms)**
2014-08-01	41,436	66,544	517,092	497	0.0961	1,372
2014-09-01	41,694	66,995	522,741	502	0.0960	1,472
2014-10-01	41,867	67,536	528,821	631	0.1193	1,455
2014-11-01	42,012	69,300	541,718	1,031	0.1903	1,497
2014-12-01	42,189	69,887	545,168	1,193	0.2188	1,425
2015-01-01	42,329	70,272	544,210	1,277	0.2347	1,510
2015-02-01	42,466	70,724	546,158	1,420	0.2600	1,549
2015-03-01	42,588	71,032	548,006	1,463	0.2670	1,542
2015-04-01	42,805	71,549	552,367	1,552	0.2810	1,437
2015-05-01	42,979	71,954	557,550	1,609	0.2886	1,538

Table 5 shows the numbers of identified redundant is-a relations for the 10 versions with respect to the length of the indirect path. For each version, $\mathbf{l_i}$ ($i = 2, \ldots, 7$) is the number of redundant is-a relations in length of i regarding to the indirect path. For example, in the version of 2015-05-01, there were 1,238 redundant is-a relations in length of 2 and 255 in length of 3. Most redundant is-a relations were in length of 2 or 3 regarding to the indirect path. There were only a couple of redundant is-a relations in length of 7. No redundant is-a relations exceeding length of 7 was identified.

Table 5: Numbers of redundant is-a relations in 10 different versions of Gene Ontology regarding to the length of the indirect path. $\mathbf{l_i}$ represents the number of redundant is-a relations in length of i regarding to the indirect path.

Version	$\mathbf{l_2}$	$\mathbf{l_3}$	$\mathbf{l_4}$	$\mathbf{l_5}$	$\mathbf{l_6}$	$\mathbf{l_7}$	**Total**
2014-08-01	421	40	23	11	1	1	497
2014-09-01	419	44	24	13	1	1	502
2014-10-01	512	72	29	15	2	1	631
2014-11-01	771	164	64	27	4	1	1,031
2014-12-01	921	174	63	27	7	1	1,193
2015-01-01	980	202	62	24	8	1	1,277
2015-02-01	1,098	220	68	24	8	2	1,420
2015-03-01	1,119	237	72	25	8	2	1,463
2015-04-01	1,198	238	78	29	7	2	1,552
2015-05-01	1,238	255	78	29	7	2	1,609

4.4 Evaluation

Even though in most cases redundant edges should be removed, in some cases the redundancy is caused by a mistake of an edge along the indirect path. For example, in Fig. 5, the assertion that "Bilateral congenital dislocation of hip" is-a "Congenital dislocation of right hip" is most likely in error. This is because a concept involving "bilateral" should not be a subclass of a concept of limited laterality: "right" (but not "left"). Removing this edge would have automatically eliminated the redundancy of the detected relation.

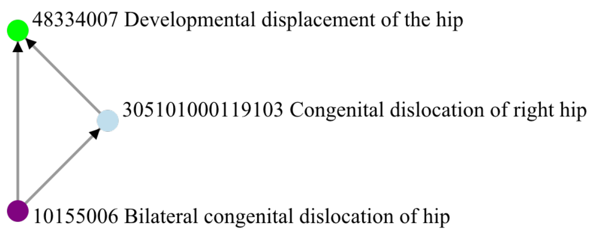

48334007 Developmental displacement of the hip
305101000119103 Congenital dislocation of right hip
10155006 Bilateral congenital dislocation of hip

Fig. 5: A visualized example of redundant is-a relation in SNOMED CT.

To evaluate the performance of FEDRR's detection of redundant is-a relations, a random sample of 30 redundant relations from SNOMED CT (2015-03-01 version) and 50 from GO (2015-05-01 version) were selected and manually reviewed by two human annotators. One annotator was asked to manually verify if the redundant hierarchical relations identified by FEDRR are correct. The other annotator was asked to review each redundant relation and provide on feedback if the redundant relation (direct edge) should be removed or an edge in the indirect path should be removed.

The first annotator verified that all of the redundant hierarchical relations identified by FEDRR are correct, that is, 100% accurancy. Table 6 shows the feedback of the second annotator. Among 30 redundant is-a relations in SNOMED CT, 24 (80%) should have direct edge removed, and 6 (20%) should have indirect edge removed. Among 50 redundant is-a relations in GO, 45 (90%) should have direct edge removed, and 5 (10%) should have indirect edge removed.

Table 6: Numbers of direct edge and indirect edge that should be removed for 30 redundant is-a relations in SNOMED CT and 50 in Gene Ontology.

	Remove **direct** edge	Remove **indirect** edge
SNOMED CT	24 (80%)	6 (20%)
Gene Ontology	45 (90%)	5 (10%)

5 Discussions

5.1 Related Work

There has been related work on exploring redundant relations in biomedical ontologies or terminologies [22–24]. Bodenreider [22] investigated the redundancy of hierarchical relations across biomedical terminologies in the Unified Medical language System. Different from this work, FEDRR focuses on developing a fast and scalable approach to detect redundant hierarchical relations in a single ontology.

Gu et al [23] investigated five categories of possibly incorrect relationship assignment including redundant relations in the Foundational Model of Anatomy. The redundant relations were detected based on the interplay between the *is_a* and other structural relationships (*part_of*, *tributary_of*, *branch_of*). A review of 20 samples from possible redundant part_of relations validated 14 errors, a 70% correctness. FEDRR differs from this work in two ways. Firstly, FEDRR aims to provide an efficient algorithm to identify redundant hierarchical relations from large ontologies with 100% accuracy. Secondly, FEDRR can be used for detecting redundant relations in all DAG with the transitivity property.

Mougin [24] studied redundant relations as well as missing relations in GO. The identification of redundant relations was based on the combination of relationships including *is_a* and *is_a*, *is_a* and *part_of*, *part_of* and *part_of*, and *is_a* and *positively_regulates*. FEDRR's main focus is to provide a generalizable and efficient approach to detecting redundant hierarchical relations in any ontology, which has been illustrated by applying it to two of the largest biomedical ontologies SNOMED CT and GO. Moreover, the redundant hierarchical relations detected by FEDRR were evaluated by human experts, while only number of redundant relations was reported in [24] without human annotator's validation.

6 Conclusion

Detecting and removing redundant relations is an important quality improvement task for biomedical ontologies because non-redundancy is the basic premise of all semantic measures derived from ontological structures, such as semantic distance between concepts and ontology mapping and alignment. We introduced FEDRR for fast and exhaustive detection of all redundant hierarchical relations in ontological hierarchies. Our algorithm runs in linear time to the size of the ontological structure in practice.

Using FEDRR, we performed systematic and exhaustive search of all redundant relations in two of the largest ontological systems in biomedicine: SNOMED CT and Gene Ontology. The algorithmic core of FEDRR is easy to implement and extremely efficient. In our extensive experiments on real-world, largest ontological structures, it took less than 20 seconds for FEDRR to process SNOMED CT and Gene Ontology.

With these results, we believe that FEDRR is production ready. After creating a user guide and a technical guide, with an associated visualization interface, we intend to release it as an open-source tool to the ontological engineering community in the near future.

Acknowledgments. We thank Shiqiang Tao for the SVG template which helped the rendering of the diagrams used for this work. The project described was supported by the National Center for Advancing Translational Sciences, UL1TR000117, and in part by the Software Solutions of Applied Research and Technology Program at Western Kentucky University. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

References

1. Cui L, Tao S, Zhang GQ. A Semantic-based Approach for Exploring Consumer Health Questions Using UMLS. AMIA Annual Symp Proc 2014, pp. 432-441.

2. Zhang GQ, Cui L, Lhatoo S, Schuele S, Sahoo S. MEDCIS: Multi-Modality Epilepsy Data Capture and Integration System. AMIA Annual Symp Proc 2014, pp. 1248-1257.
3. Jayapandian C, 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, Atlanta, 2014 (in press).
4. 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.
5. Couto, Francisco M., Mário J. Silva, and Pedro M. Coutinho. Measuring semantic similarity between Gene Ontology terms. Data & knowledge engineering 61, no. 1 (2007): 137-152.
6. Giunchiglia F, Autayeu A, Pane J, S-match: an open source framework for matching lightweight ontologies Semantic Web, 3 (3) (2012), pp. 307-317
7. Donnelly K. SNOMED-CT: The advanced terminology and coding system for eHealth. Stud Health Technol Inform Vol. 121, pages 279-90, 2006.
8. Bodenreider, O. The unified medical language system (UMLS): integrating biomedical terminology. Nucleic acids research, 32(suppl 1), D267-D270, 2004.
9. Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. Nucleic acids research 32.suppl 1: D258-D261, 2004.
10. Min H, Perl Y, Chen Y, Halper M, Geller J, Wang Y. Auditing as part of the terminology design life cycle. J Am Med Inform Assoc 2006;13(6): 676-690.
11. He Z, Ochs C, Agrawal A, Perl Y, Zeginis D, Tarabanis K, Elhanan G, Halper M, Noy N, Geller J. A family-based framework for supporting quality assurance of biomedical ontologies in BioPortal. In AMIA Annual Symposium Proceedings 2013, pp. 581-590.
12. Tao S, Cui L, Zhu W, Sun M, Bodenreider O, Zhang GQ. Mining Relation Reversals in the Evolution of SNOMED CT Using MapReduce. AMIA Joint Summits on Translational Science 2015, pp. 46-50.
13. Ceusters W. Applying Evolutionary Terminology Auditing to SNOMED CT. AMIA Annu Symp Proc. 2010 Nov 13;2010:96-100.
14. Hartung M, Grob A, Rahm E. COnto-Diff: generation of complex evolution mappings for life science ontologies. J Biomed Inform. 2013 Feb;46(1):15-32.
15. Kirsten T, Gross A, Hartung M, Rahm E. GOMMA: a component-based infrastructure for managing and analyzing life science ontologies and their evolution. J Biomed Semantics. 2011 Sep 13;2:6. doi: 10.1186/2041-1480-2-6.
16. Jiang G, Chute CG. Auditing the semantic completeness of SNOMED CT using formal concept analysis. J Am Med Inform Assoc 2009;16(1):89-102.
17. 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, 2010.
18. Zhang GQ and Bodenreider O. Large-scale, exhaustive lattice-based structural auditing of SNOMED CT. American Medical Informatics Association (AMIA) Annual Symposium, pages 922-926, 2010.
19. Zhang GQ, Zhu W, Sun M, Tao S, Bodenreider O, Cui L. MaPLE: A MapReduce Pipeline for Lattice-based Evaluation of SNOMED CT. IEEE International Conference on Big Data, 2014;754-9.
20. Sedgewick R, Wayne K, Algorithms (4th ed.), Addison-Wesley Professional, pp. 661-666, ISBN 9780321573513, 2011.
21. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001), Section 22.4: Topological sort, Introduction to Algorithms (2nd ed.), MIT Press and McGraw-Hill, pp. 549-552, ISBN 0-262-03293-7.
22. Bodenreider O. Strength in numbers: exploring redundancy in hierarchical relations across biomedical terminologies. AMIA Annual Symp Proc 2003, pp. 101-105.
23. Gu HH, Wei D, Mejino JLV, and Elhanan G. Relationship auditing of the FMA ontology. Journal of biomedical informatics 42(3): 550-557, 2009.
24. Mougin F. Identifying redundant and missing relations in the gene ontology. Studies in health technology and informatics 210: 195-199, 2014.