Semantic Web 0 (0) 1–28
IOS Press

Discovery of Emerging Design Patterns in Ontologies Using Tree Mining

Editor(s): Rinke Hoekstra, Vrije Universiteit Amsterdam, Netherlands
Solicited review(s): Vojtěch Svátek, University of Economics, Prague, Czech Republic; Eva Blomqvist, Linköpings Universitet, Sweden; Heiko Paulheim, Universität Mannheim, Germany

Agnieszka Ławrynowicz [a,*], Jedrzej Potoniec [a,**], Michał Robaczyk [a], Tania Tudorache [b]
[a] *Faculty of Computing, Poznan University of Technology, ul. Piotrowo 3, 60-965 Poznan, Poland*
[b] *Stanford Center for Biomedical Informatics Research, Stanford University, 1265 Welch Road, Stanford, CA 94305, USA*

Abstract. The research goal of this work is to investigate modeling patterns that recur in ontologies. Such patterns may originate from certain design solutions, and they may possibly indicate emerging ontology design patterns. We describe our tree-mining method for identifying the emerging design patterns. The method works in two steps: (1) we transform the ontology axioms in a tree shape in order to find axiom patterns; and then, (2) we use association analysis to mine co-occuring axiom patterns in order to extract emerging design patterns. We conduct an experimental study on a set of 331 ontologies from the BioPortal repository. We show that recurring axiom patterns appear across all individual ontologies, as well as across the whole set. In individual ontologies, we find frequent and non-trivial patterns with and without variables. Some of the former patterns have more than 300,000 occurrences. The longest pattern without a variable discovered from the whole ontology set has size 12, and it appears in 14 ontologies. To the best of our knowledge, this is the first method for automatic discovery of emerging design patterns in ontologies. Finally, we demonstrate that we are able to automatically detect patterns, for which we have manually confirmed that they are fragments of ontology design patterns described in the literature. Since our method is not specific to particular ontologies, we conclude that we should be able to discover new, emerging design patterns for arbitrary ontology sets.

Keywords: ontology, ontology fragment, emerging design pattern, ODP, pattern mining, tree mining, ontology reuse, BioPortal

1. Design Patterns in Ontology Engineering

Today's methodological guidelines in ontology engineering (see, for instance, the NeON methodology [33]) suggest to *reuse* existing ontologies, or their fragments, while developing a new ontology. The solutions to common modeling problems, such as, modeling partonomies, are often documented as *Ontology Design Patterns (ODP)* [28], and authors may choose to reuse them in their ontologies.

ODPs have been proposed as a method analogous to design patterns in software engineering [11,12,28] that aim to provide good quality solutions to recurring modeling problems. Blomqvist et al. [4] have proposed various types of ODPs, e.g., content, structural or lexico–syntactic ones. Ontology patterns may also be specific for a certain domain, for example, Aranguren et al. [3,10] developed ontology patterns specific for biology. Many of the proposed patterns can be found in two repositories, the ODP Portal,[1] and the Manchester ODP Catalog.[2] Some ontology editing environments (e.g., Protégé [18] and the NeOn toolkit [13]) offer functionalities to support the use of patterns in the form of wizards that help users create values

*Corresponding author. E-mail: alawrynowicz@cs.put.poznan.pl
**Corresponding author. E-mail: jpotoniec@cs.put.poznan.pl

[1] http://www.ontologydesingpatters.org/
[2] http://www.gong.manchester.ac.uk/odp/html/

partitions, value sets, or lists [9] directly as part of the ontology authoring process.

In the current practice, ontology authors create the ontology patterns manually, and sometimes, they upload them to one of the ontology patterns repositories. However, developing such patterns is very laborious. Moreover, ODP repositories are not yet comprehensive—not all recommended design solutions are recorded in these repositories of patterns. Even with the availability of such repositories, domain experts still have difficulties to find and apply a suitable modeling pattern, when having to choose among several, possibly abstract patterns.

In many cases, recurring patterns of axioms may exist in ontologies, even if they have not been officially published as a part of a recommended design pattern. We call such empirical patterns *emerging design patterns* since they are not full ODPs (yet). In addition, a full ODP usually contains an accompanying textual explanation, diagrams, usage examples, and other components. The identification of emerging design patterns may be the first step towards (semi-)automatic creation of ODPs. Recently, Blomqvist et al. [5] have identified the task of analyzing ontologies to discover such "hidden" design patterns as useful, but non-trivial, and requiring significant support.

For the purpose of this work, we will identify three different types of patterns: Axiom Patterns with Variables (APV), Axiom Patterns without Variables (APNV), and Class Frame Patterns (CFP). Figure 1 shows examples for the three types of patterns with the goal of introducing the terminology. We provide the formal definitions of the three pattern types in Section 3.2.

Our research objective in this work is to investigate patterns that occur frequently in individual ontologies, as well as in a group of ontologies, such as the ones stored in an ontology repository. Our work is guided by the following research questions:

- Do certain patterns recur in ontologies? Can we generalize over such patterns to mine more generic templates?
- Do such patterns appear within a group of ontologies?
- Do such patterns exist on the axiom level? Do they exist on the level of sets of axioms?
- Are we able to automatically detect fragments of documented ODPs?

The main contributions of our work are:

1. Axiom Pattern with Variables (APV)

$?lhs$ SubClassOf `hasTopology` some $?c$

2. Axiom Pattern without Variables (APNV)

`nucleic acid extraction` SubClassOf `has_specified_input` some (`organism` or `cultured cell population` or `sample from organism`)

3. Class Frame Pattern (CFP)

{$?lhs$ SubClassOf `hasTopology` some $?c1$,
$?lhs$ SubClassOf `isServedBy` some $?c2$}

Axiom Pattern (AP) = APV *or* APNV

Ontology Pattern (OP) = AP *or* CFP

Fig. 1. Examples for the three types of patterns covered in this paper: 1. Axiom Pattern with Variables (APV), 2. Axiom Pattern without Variables (APNV), and 3. Class Frame Pattern (CFP).

- We propose a method based on tree mining for discovering frequent *axiom patterns* in ontologies. The method operates by transforming ontology axioms into a tree form, then applying frequent tree mining, and finally decoding the frequent trees into axiom patterns.
- We propose an association–analysis method to discover frequent *class frame patterns* in ontologies (i.e., frequent axiom pattern sets) on top of the discovered axiom patterns. To the best of our knowledge, this is the first method that is able to automatically discover such type of (emerging) design patterns in ontologies.
- We conduct an experimental analysis on BioPortal ontologies with the goal to discover frequent ontology patterns.

Our analysis reveals that: (i) We are able to identify recurring patterns in ontologies, both at the axiom level, and at the level of sets of axioms; (ii) We are able to automatically extract non-trivial, and significantly–frequent patterns without variables; (iii) We are able to discover patterns with variables; and (iv) We are able to automatically mine fragments of already known ODPs. All results obtained during the experiments are available online at: `http://semantic.cs.put.poznan.pl/bioportal-patterns/`.

The rest of this paper is structured as follows. Section 2 summarizes the related work. Section 3 introduces the problem of tree mining, and formally de-

fines the notions of ontology axiom and class frame patterns. Section 4 describes the BioPortal dataset used in this study, and the proposed methods. Section 5 describes the results of our experiments. We discuss our results in Section 6, and conclude in Section 7.

2. Related work

Prior research on the topic of extracting ontology patterns has been quite scarce. Some previous works dealt with studying the syntactic properties of OWL ontologies on the Web. In their work, Wang et al. [39] presented statistics on the occurrence frequency of OWL language constructs, and the structure of ontology class hierarchies, in a corpus of ontologies. Zamazal et al. [34] conducted a study on collections of OWL ontologies with the aim of determining the frequency of several combined name and graph patterns, which potentially indicate underlying structural clusters. These works mainly deal with lexical patterns, and do not tackle mining recurring fragments in ontologies.

Mortensen et al. [23] studied the use of ODPs in BioPortal ontologies. The authors encoded 68 ODPs from two online pattern libraries (Manchester ODP Catalog, and the ODP Portal) using the Ontology Pre-Processor Language (OPPL).[3] The goal of their work was to determine how prevalent ODPs are in BioPortal ontologies. This study only considered structural and content ODPs, while omitting other types of patterns. After filtering out patterns that were undetectable, trivially supported, poorly reviewed, and whose properties were not present in the ontologies, they found that only 14 patterns are reused in the BioPortal repository.

In our previous work [19], we presented a method, Fr-ONT, for mining patterns in ontologies. However, our data–driven method worked only on ontologies with instances. The method iteratively constructed new ontology classes, and checked their frequency in terms of the number of instances rather than mining frequent fragments in existing axioms, as we do in our current work.

In their work, Khan and Blomqvist [17] detected content ODPs in existing ontologies. Their method works top-down, starting from existing ODPs and trying to find their instantiations in an ontology. In our method, we do not require an ODP as an input, and we mine (possibly new) patterns bottom-up. Similarly, Šváb-Zamazal et al. [35] considered the problem of detecting (logical) patterns top-down, starting from a particular pattern. Thörn et al [37] studied potentials and limits of graph-algorithms for discovering ontology patterns based on a definition of what structures are considered patterns. Their conclusion was that graph-pattern algorithms appear inefficient for finding patterns in ontologies. Tempich and Volz [36] performed a statistical analysis of the DAML ontology library. They mostly studied the language primitives with a goal to establish a benchmark for Semantic Web reasoners.

The approach taken by Mikroyannidi et al. [21,22] with the Regularities Inspector for Ontologies (RIO) is the closest to our approach. The authors used clustering to identify regularities in the usage of entities in axioms within an ontology. The authors defined the distance based on the similarity of the structure of ontology axioms [21]. Thus, the process of clustering groups axioms (more precisely, axiom templates) based on their similar structure. Our work differs from this approach in two important aspects. First, the method that we use is different, and is based on frequency- and association analyses. Second, we are able to discover sets of axiom templates (class–frame fragments), rather than only single–axiom templates, due to the use of association analysis, which can group axiom templates of very different structures. We discuss our approach in comparison to RIO in more detail in Section 6.5.

3. Preliminaries

3.1. Tree mining

Tree mining is an area of data mining that deals with the discovery of frequent subtrees in tree–shaped data structures. Tree mining has been applied to several areas, such as, bioinformatics, web usage mining, and mining XML files [42]. We use the *SLEUTH* algorithm [42]—an extended version of the *TreeMiner* algorithm [43]—to discover frequent patterns in ontologies.

In the following paragraphs, we will introduce some basic definitions and terminology.

- A *tree* is a directed, connected, acyclic graph (V, E), where V is a set of nodes, and $E \subset V \times V$ is a set of edges.
- A *rooted tree* is a tree with a distinguished *root* node.

[3] http://oppl2.sourceforge.net

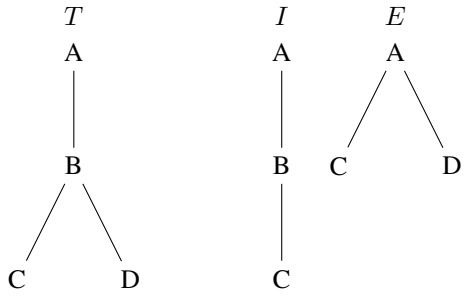


Fig. 2. T is a labeled tree. I is an induced subtree of T, i.e. it contains all edges between the nodes of I, which were present in T. E is an embedded subtree of T, i.e. some paths between nodes of E, which were present in T, are represented as edges.

- A *labeled tree* is a triple (V, E, l), where (V, E) is a tree and $l : V \to L$ is a labeling function mapping every node to some label from the set L.
- A *path* is a sequence of nodes $(n_1, n_2, \ldots, n_k)$ such that $(n_i, n_{i+1}) \in E$ for all $i \in \{1, 2 \ldots, k-1\}$.
- A *forest* is a set of rooted trees and a *labeled forest* is a set of rooted, labeled trees.
- An *induced subtree* of a rooted labeled tree $T = (V_T, E_T, l_T)$ is a labeled tree $S = (V_S, E_S, l_S)$, such that $V_S \subseteq V_T$, $E_S = (V_S \times V_S) \cap E_T$, i.e., E_S consists of all edges between the nodes of V_S in the tree T, and $l_S(n) = l_T(n)$ for every node $n \in V_S$.
- An *embedded subtree* is a generalization of an induced subtree, such that $(w, v) \in E_S$, iff w is on a path from the root of T to v. A sample induced subtree and a sample embedded subtree are presented in Figure 2.
- A *parent* of a node n is a node m, such that $(n, m) \in E$, and m immediately precedes n on a path from the root to n; n is called a *child* of m.

In this work, we assume that all trees are rooted and labeled, and that all forests are labeled.

By the *support* of a tree S over a forest F, we understand a value

$$\sigma_F(S) = \sum_{T \in F} d(S, T) \tag{1}$$

where $d(S, T) = 1$, if S is an induced subtree of T, and $d(T, S) = 0$ otherwise. The *relative support* of a tree S over a forest F is $\sigma_F^r(S) = \frac{\sigma_F(S)}{|F|}$, which is the support of S over F divided by the number of trees in the forest F. A *frequent subtree* S of a given forest F is a tree such that its support $\sigma_F(S)$ is greater than a given threshold.

The *tree mining problem* can be defined in many different ways. For the purpose of this work, our aim is to enumerate all frequent subtrees of a given forest. The interested reader is referred to the work of Zaki [42] for different possible formulations of the problem (e.g., mining of ordered trees or different support definitions).

3.2. Ontology patterns

Our research is concerned with identifying patterns in ontologies represented in the *Web Ontology Language (OWL)* [27]. In this subsection, we introduce briefly OWL and the terminology used throughout this paper.

An *OWL ontology* is a set of *axioms*. The axioms are constructed from *entities*, and various *constructors* (e.g., logical operators).

Entities are the basic building blocks of OWL ontologies, defining the vocabulary of an ontology. An OWL vocabulary $N_{\mathcal{O}}$= (N_C, N_{OP}, N_{DP}, N_{AP}, N_I, N_D, N_{LIT}) is a 7-tuple where N_C is the set of class names (atomic class expressions), N_{OP} is the set of object property names, N_{DP} is the set of data property names, N_{AP} is the set of annotation property names, N_I is the set of individual names, N_D is the set of datatype names, and N_{LIT} is the set of well-formed literals.

OWL provides several constructors to combine entities into more complex class expressions. The complex class expressions are defined inductively using the following grammar:[4]

$$\begin{aligned} C \to\ & A \mid \mathsf{not}\, C \mid C_1 \,\mathsf{and}\, \ldots\, C_n \mid C_1 \,\mathsf{or}\, \ldots\, C_n \mid \\ & \{a\} \mid p \,\mathsf{some}\, C \mid p \,\mathsf{only}\, C \mid p \,\mathsf{min}\, n \mid p \,\mathsf{max}\, n \mid \\ & p \,\mathsf{min}\, n\, C \mid p \,\mathsf{max}\, n\, C \mid p \,\mathsf{exactly}\, n \mid \\ & p \,\mathsf{exactly}\, n\, C \mid p \,\mathsf{value}\, a \mid t \,\mathsf{some}\, D \mid t \,\mathsf{only}\, D \mid \\ & t \,\mathsf{min}\, n \mid t \,\mathsf{max}\, n \mid t \,\mathsf{min}\, n\, D \mid t \,\mathsf{max}\, n\, D \mid \\ & t \,\mathsf{exactly}\, n \mid t \,\mathsf{exactly}\, n\, D \mid t \,\mathsf{value}\, \mathrm{lit} \end{aligned} \tag{2}$$

C stands for (possibly complex) class expression, $A \in N_C$, $a \in N_I$, $p \in N_{OP}$, $q \in N_{OP}$, $t \in N_{DP}$, $D \in N_D$, n is a non-negative integer, and lit $\in N_{LIT}$ is a

[4] We use the Manchester syntax [15] for OWL ontologies throughout the paper

literal. By N_{CC} we denote the set of class constructors: not, and, or, some, only, min, max, exactly, value.

For the analyses described in this paper, we consider two classes of logical axioms, namely subclass axioms C_1 SubClassOf C_2, and equivalent class axioms C_1 EquivalentTo C_2. We omit non-logical axioms (i.e., axioms that are not used by a reasoner for inference, such as annotation axioms). Furthermore, we only consider axioms having a named class on their left-hand side (*lhs*), i.e., $C_1 \in N_C$ in our case. This restriction is motivated by a common ontology engineering practice, in which one concentrates on modeling sets of descriptions of entities, rather than sets of arbitrary axioms. Most ontology editing environments, such as Protégé, support this practice via an entity–centric interface.

Following the terminology of Horridge et al. [16], we define the *class frame* of a class A w.r.t. $\mathcal{O}$ as the subset–maximal set of axioms $CF_A \subseteq \mathcal{O}$ where each axiom in CF_A has one of the forms:

$$A \text{ SubClassOf } C$$
$$A \text{ EquivalentTo } C$$

In other words, a class frame CF_A for class A in an ontology $\mathcal{O}$ contains all the subclass and equivalent axioms from that ontology, in which the class A appears on the left-hand side of the axioms. The right-hand side of the axiom may contain any arbitarily complex class expression. Table 1 shows a simple anatomical ontology describing internal organs (inspired by the medical ontology GALEN [29]), which contains eleven axioms in two ontologies. The example illustrates four class frames, which are comprised from axioms 1-5, 6-7, 8-9 and 10-11, respectively.

As part of this work, we also identify patterns containing variables. We define the following sets of symbols (variable names) that are not in the vocabulary of $\mathcal{O}$, $N_{\mathcal{O}}$: $X = (X_{CC}, X_C, X_D, X_{OP}, X_{DP}, X_I, X_{LIT}, X_n, X_f)$, where each variable $?classexpr \in X_{CC}$ may be bound only to an OWL class expression, each variable $?c \in X_C$ to a symbol from N_C, each variable $?datatype \in X_D$ to a symbol from N_D, each variable $?op \in X_{OP}$ to a symbol from N_{OP}, each variable $?dp \in X_{DP}$ to a symbol from N_{DP}, each variable $?i \in X_I$ to a symbol from N_I, each variable $?literal \in X_{LIT}$ to a symbol from N_{LIT}, each variable $?cardinality \in X_n$ to a non-negative integer, and each variable $?facet \in X_f$ may be bound only to a datatype restriction. Please note that when multiple variables of the same type appear in a single pattern, they would be extended with consecutive natural numbers ($?classexpr1$ etc.). Moreover, we denote a variable appearing on the left-hand side of an axiom fragment as $?lhs$.

Table 1
Simple ontology on anatomy serving for the illustration purposes.

No.	Axiom
	Ontology1
1	`Heart` SubClassOf `hasTopology` only `Tubular`
2	`Heart` SubClassOf (`hasTopology` some `Tubular`) and `InternalOrgan`
3	`Heart` SubClassOf `hasFeature` some `Tubular`
4	`Heart` SubClassOf `hasFeature` some (not `Tubular`)
5	`Heart` SubClassOf `hasMass` exactly `1 xsd:float`
6	`Tubular` EquivalentTo `Topology` and (`hasState` some `TubularSt`)
7	`Tubular` SubClassOf `hasState` some `TubularSt`
	Ontology2
8	`Kidney` SubClassOf `hasTopology` some `Solid`
9	`Kidney` SubClassOf `isServedBy` some `RenalAnteriorSegmentalArtery`
10	`Liver` SubClassOf `hasTopology` some `Solid`
11	`Liver` SubClassOf `isServedBy` some `HepaticVein`

Definition 3.1 *An* **axiom pattern with variables** *(APV) of an OWL axiom α, Q^α, with respect to ontology $\mathcal{O}$ is obtained by replacing $n > 0$ elements of α from $N_{\mathcal{O}}$ with elements from X.*

A sample axiom pattern with variables (APV) corresponding to our illustrative example from Table 1 is:

$?lhs$ SubClassOf `hasTopology` some $?c$

Definition 3.2 *An* **axiom pattern without variables** *(APNV), a.k.a.* **axiom fragment***, of an OWL axiom α, Q^α, with respect to ontology $\mathcal{O}$ is obtained by removing a part of α. In a special case, Q^α may be equal to α.*

An example of an axiom pattern without variables is the following (we replaced the entity ids with their labels to improve readability. See details in Appendix 10):

Definition 3.3 *A* **class frame pattern** *(CFP) of the class frame CF_A of an ontology class A w.r.t. $\mathcal{O}$ is the set of axiom patterns $\mathbb{Q}_A^{CF}$ where each pattern in $\mathbb{Q}_A^{CF}$ has the left-hand side of either of the form: a named class A or $?lhs$.*

nucleic acid extraction SubClassOf has_specified_input some (organism or cultured cell population or sample from organism)

{?lhs SubClassOf hasTopology some ?$c1$, ?lhs SubClassOf isServedBy some ?$c2$}

A sample class frame pattern (CFP) with a variable left-hand side is:

Definition 3.4 *An* **axiom pattern** *(AP) is either an axiom pattern with variables (APV) or an axiom pattern without variables (APNV).*

Definition 3.5 *An* ***ontology pattern*** *(OP), or a* **pattern** *for short, is either an axiom pattern (AP) or a class frame pattern (CFA).*

The examples for the different types of patterns are also shown in Figure 1.

4. Material and methods

4.1. BioPortal ontologies

For the experimental evaluation, we downloaded on July 25, 2015 a snapshot of all ontologies from the BioPortal ontology repository[5] [40], using the BioPortal API. We obtained 442 ontology files, 34 of which turned out to be empty (e.g., due to licence restrictions, like in the case of SNOMED CT, or due to errors in the uploaded files, as in the case of AAO). As these files came in different formats (e.g., RDF/XML, OWL/XML, OBO), we used Robot[6] from OntoDev[7] to convert all ontologies to RDF/XML format. Further, using the OWL API[8] [14], we extracted axioms relevant for this work, i.e. SubClassOf and EquivalentTo axioms, and converted them to trees and forests, as described in Section 4.2 (i.e., we converted each axiom to a tree, and each ontology to a forest of trees). We used only axioms defined in the ontologies themselves, ignoring all `owl:import` statements.

4.2. Encoding of an ontology to a forest

Our aim is to find frequent patterns based on OWL subclass and equivalent class axioms, such that their left-hand side is a named class. We convert every axiom to a single tree, which then is used as an input for the SLEUTH algorithm (Section 4.3). We build the tree by recursively processing the arguments of each constructor, starting with SubClassOf or EquivalentTo. Each constructor, or named object corresponds to a node, and every node is labeled with a pair, defining the type of the node, and its name. We define 10 types of the nodes, which allow us to preserve the OWL semantics of the names:

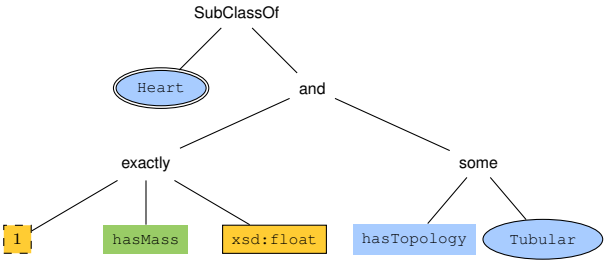


Fig. 3. Tree representation of an axiom: `Heart` SubClassOf (`hasTopology` some `Tubular`) and (`hasMass` exactly 1 `xsd:float`). Shapes correspond to the types of the nodes: class constructor nodes have no outline and no background, an ellipse stands for a named class and a double ellipse is for left-hand side, a rectangle with no outline stands for a property, a rectangle with a dashed outline is for a cardinality value and with a solid outline is for a datatype.

class constructor: OWL constructor concerning classes or datatypes (e.g., intersection), corresponding to N_{CC};
class, datatype: named class, named datatype, corresponding to N_C and N_D, respectively;
object property, datatype property: named object or datatype properties, corresponding to N_{OP} and N_{DP}, respectively;
individual, literal: named individual, literal value (e.g., in *hasValue* restrictions), corresponding to N_I and N_{LIT}, respectively;
cardinality: cardinality values in cardinality restrictions, corresponding to n;
facet: facet datatype restriction (e.g., in limiting range of integers);
left-hand side: left-hand side of subclass axioms (always a named class);

As an example, let us consider the following axiom:

`Heart` SubClassOf (`hasTopology` some `Tubular`) and (`hasMass` exactly 1 `xsd:float`)

The tree corresponding to the axiom is presented in Figure 3.

[5] http://bioportal.bioontology.org/
[6] https://github.com/ontodev/robot
[7] http://ontodev.com/
[8] http://owlapi.sourceforge.net/

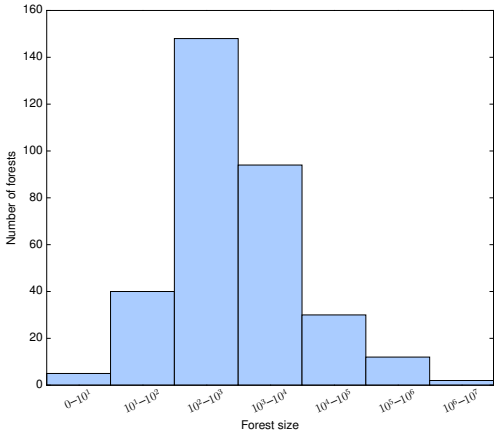


Fig. 4. Forests' sizes histogram. The forests were obtained from 331 BioPortal ontologies, each converted to a forest of trees, where each tree corresponds to a SubClassOf or EquivalentTo axiom.

Because SLEUTH operates only on numeric labels, every distinct pair (type and name) is assigned an unique integer value. These values are stored to decode frequent trees back.

By applying this method to the BioPortal ontologies, we obtained a set of 331 non-empty forests, the smallest of which containing 4 trees, while the largest containing 1,833,925 trees. The histogram of the forests' sizes is presented in Figure 4. The median size of a forest was 629, while the average size was 25,308.4, with the standard deviation of 147,725.3.

4.3. SLEUTH

In order to extend and apply SLEUTH [42]—an efficient algorithm for mining frequent, unordered, embedded subtrees—to our use case, we needed to encode each tree into an efficient string representation, as described in this work [43]. Precisely, a tree T is traversed with a depth–first preorder manner, and the labels of visited nodes are stored in the string. Every time the backtracking is performed, we add a special symbol to it, namely the dollar sign, '$'. An example is presented in Figures 5a and 5b.

The *SLEUTH* algorithm is based on an observation that every tree can be constructed by a sequence of operations, each consisting of adding a new node as a child of an existing one, in a such way that the new node is the last node in the depth–first preorder labeling. Let k–subtree be a subtree containing exactly k nodes. Given a frequent $(k-1)$–subtree, SLEUTH algorithm constructs all frequent k–subtrees, which differ from the original subtree only by the last node in the depth–first preorder. To guide this construction, only nodes that were used in the previous step are considered, i.e., nodes that were used to extend a $(k-2)$–subtree to a forest of $(k-1)$–subtrees. In order to make the computations faster, a tree representation called *scope–list* is used. Using the scope–lists, we can verify in constant time, if given a tree and a $(k-1)$–subtree in this tree, whether the k–subtree also occurs in the tree. An example of such representation is presented in Figures 5c, and 5d. Every frequent k–subtree found this way is then recursively used to find frequent $(k+1)$–subtrees.

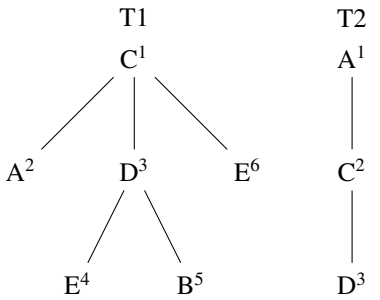


Fig. 5a. A forest of two trees, T1 rooted in C and T2 rooted in A. Numbers in the superscript represent an order of depth–first preorder traversal of the trees.

T1: (C A $ D E $ B $ $ E $)
T2: (A C D $ $)

Fig. 5b. Depth–first preorder string encoding of the trees T1 and T2. $ is the special symbol to indicate backtracking.

A	B	C	D	E
T1 $[2,2]$	T1 $[5,5]$	T1 $[1,6]$	T1 $[3,5]$	T1 $[4,4]$
T2 $[1,3]$		T2 $[2,3]$	T2 $[3,3]$	T1 $[6,6]$

Fig. 5c. Scope–list representation of the trees T1 and T2. $[a,b]$ is a node scope for a given node, where a is the position of the node in the depth–first preorder traversal and b is the position of the last descendant in the same traversal (or a if the node is a leaf).

(C D $)	(C E $)	(C E $ E $)
T1 (1) $[3,5]$	T1 (1) $[4,4]$	T1 $(1,4)$ $[6,6]$
T2 (2) $[3,3]$	T1 (1) $[6,6]$	

Fig. 5d. Scope–list representations for some more complex subtrees. Values in parentheses are *match labels* that is a proof made of nodes' positions that given subtree (apart of the last node) indeed exists in a particular tree. The node scopes are scopes for the last nodes.

4.4. FF-SLEUTH

Despite its feasibility, the SLEUTH algorithm has two disadvantages, which renders it unsuitable for our use case. The first disadvantage is concerned with the embedded patterns which do not provide valuable information about axioms in an ontology. For example, consider these two axioms, corresponding to the axioms #1 and #4 in Table 1:

(1) `Heart` SubClassOf: `hasTopology` only `Tubular`
(2) `Heart` SubClassOf: `hasFeature` some not `Tubular`

The trees corresponding to these axioms are presented in Figure 6. One of embedded patterns occurring in the axioms is `Heart` SubClassOf `Tubular`, yet such a pattern is not justified by the original axioms, as in both of the axioms `Heart` is related to `Tubular` topology via some property.

The second disadvantage of the SLEUTH algorithm is that it mines tree patterns in a single forest, which is not sufficient. Indeed, a single ontology forms already a forest, and we are also interested in mining patterns that occur in multiple ontologies. Thus, we need a method to mine tree patterns in a family of forests.

To address these two disadvantages, we modified the SLEUTH algorithm. To deal with the first issue (unintended patterns), we keep and extend only the induced subtrees. Other subtrees—those that are embedded, but are not induced—are discarded. This change does not affect the soundness of the algorithm, as all induced subtrees of a frequent, induced subtree must also be frequent, and are therefore constructed as well.

We addressed the second issue (mining in a family of forests) by introducing a new measure of support.

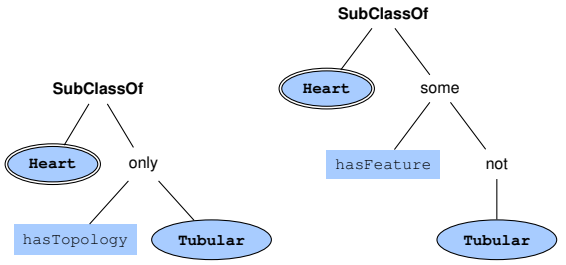


Fig. 6. Trees for axioms [`Heart` SubClassOf `hasTopology` only `Tubular`], and [`Heart` SubClassOf `hasFeature` some not `Tubular`] (resp. the axioms #1 and #4 in Table 1). Bold symbols denote nodes occurring in both trees and constituting an embedded subtree corresponding to the axiom [`Heart` SubClassOf `Tubular`].

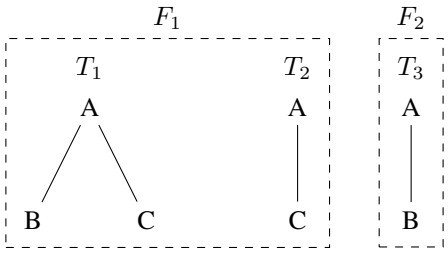


Fig. 7. A family of two forests $\mathcal{F} = \{F_1, F_2\}$ consisting of three trees: T_1, T_2, T_3. Subtrees (A) and (A B $), using the string representation, have support 2 as they occur in at least one tree in both forests. Subtree (A C $) has support 1 even though it occurs in two trees T_1 and T_2, because they both belong to the same forest F_1.

Considering a family of forests $\mathcal{F}$, we define the support of a subtree in a family of forests as the number of forests containing at least one tree containing the given subtree, i.e.,

$$\sigma_{\mathcal{F}}(S) = \sum_{F \in \mathcal{F}} \operatorname{sgn}\left(\sum_{T \in F} d(S,T)\right) \tag{3}$$

Figure 7 shows an example explaining how this new support is computed. The support is also applied to the axiom pattern corresponding to the tree after decoding it. To better exemplify how the support is computed, consider the two ontologies shown in Table 1 and the axiom pattern with variables (APV):

$?lhs$ SubClassOf `hasTopology` some $?c$

Recall the example from Table 1. This axiom patterns has support 1 in Ontology 1 (matches axiom #2), and support 2 in Ontology 2 (matches the axioms #8 and #10). The support of the axiom in the family of trees (composed of Ontology 1 and 2) is 2, as it matches at least one axiom from each of the ontologies.

As the original SLEUTH implementation[9] is highly optimized and complex, we decided to reimplement SLEUTH in Java. We developed FF-SLEUTH (Family of Forests SLEUTH), a Java implementation of the SLEUTH algorithm with the above–mentioned modifications, which is available in the Git repository: `https://bitbucket.org/leolod/jsleuth`

[9] Available at: `http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/Software#toc15`

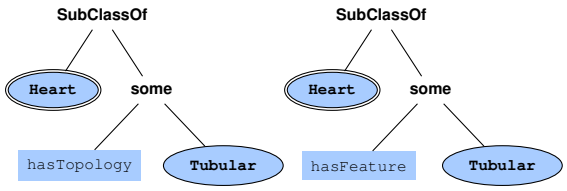


Fig. 8. There is one maximal frequent subtree of these two trees, with nodes denoted by bold symbols. All subtrees of this maximal tree are also frequent, but are not useful for our analysis.

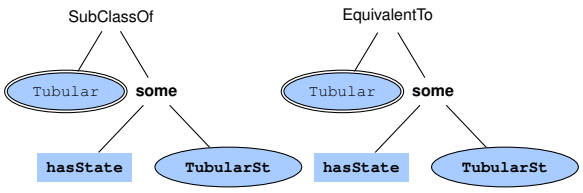


Fig. 9. During mining we also discover frequent subtrees that do not contain SubClassOf or EquivalentTo, yet we discard them, as they could lead us to wrong conclusions about axioms in mined ontology.

4.5. *Decoding a frequent subtree to a frequent axiom pattern*

Both SLEUTH and FF-SLEUTH compute the set of all frequent induced subtrees. Obviously, not all of the computed subtrees are useful for our purposes mainly due to two reasons.

First, we are interested only in maximal frequent subtrees—frequent subtrees for which none of their proper supertrees is frequent. Consider the forest in Figure 8: there are many frequent subtrees of these two trees, such as, (SubClassOf) or (SubClassOf `Heart` $), encoded in the string representation presented in Section 4.3. Obviously, these are just subtrees of a maximal subtree hidden there: (SubClassOf `Heart` $ some `Tubular` $ $).

The second reason is that we aim to mine axiom patterns with or without variables. Therefore, we will consider only the frequent subtrees which contain SubClassOf or EquivalentTo nodes. The line of reasoning here is similar to the one against embedded subtrees, presented in the Section 4.4. Consider the forest in Figure 9. One of the maximal frequent subtrees is (some `hasState` $ `TubularState` $). Yet, such a subtree does not constitute an axiom pattern. Indeed, we cannot know, if a class expression corresponding to this frequent subtree is one of the operands of SubClassOf or EquivalentTo, or maybe it is nested somewhere deeper, such as in an expression not (`hasState` some `TubularSt`).

For the further processing, we will use only the frequent subtrees which fulfill both of the above–mentioned criteria—being frequent and having the proper form. The next step in our analysis is to transform the mined trees back into *frequent axiom patterns*, which involves two steps. In the first step, we decode labels back from their numerical form to the pairs using the stored information (see Section 4.2). In this way, we obtain a (possibly incomplete) tree representations for some axioms. An example of such a tree is presented in Figure 10. In the second step, the tree is

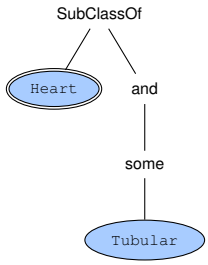


Fig. 10. A frequent subtree with decoded labels that is an incomplete tree representation of some axiom and that (after completion) will form a frequent axiom pattern `Heart` SubClassOf (?*op* some `Tubular`) and ?$classexpr$.

completed by inserting a minimal number of variables, such that the tree would correspond to a frequent axiom pattern. For example, the tree in Figure 10 contains a constructor some which requires a property, and a class expression, or a datatype. There is a class expression `Tubular`, so an object property is missing. Also, clearly the and expression is incomplete, at least missing the second operand. It is not clear (in general) how many operands we should add there, so we add a minimal number, which is one.

After the completion, we obtain a frequent axiom pattern: `Heart` SubClassOf (?*op* some `Tubular`) and ?$classexpr$. We favor object properties and class constructors over datatype properties and datatypes, e.g., in rare cases, when there are no children for a node labeled some, we add variables for an object property, and a class constructor, instead of variables for a datatype property and a datatype.

The complete algorithm for decoding and completion is published as a supplementary material[10].

[10] `https://semantic.cs.put.poznan.pl/bioportal-patterns/string-to-manchester.pdf`

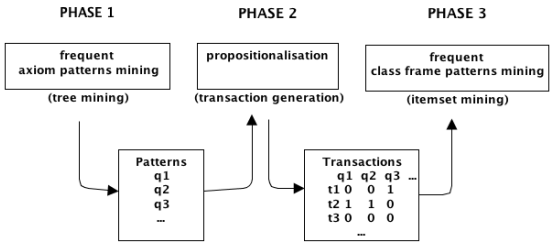


Fig. 11. A three-phase process: mining frequent class frame patterns on top of the discovered frequent axiom patterns with use of propositionalisation.

4.6. Mining class frame patterns

So far, we have described the methods for computing single axiom patterns. In this section, we describe how to compute class frame patterns (CFP) based on the discovered axiom patterns. In the simplest case, there might be axiom patterns that have a named class on their left–hand side (*lhs*). For this case, it is sufficient to group the discovered axioms that share the same *lhs*. However, there are cases in which the axiom patterns have a variable on their *lhs*. For this latter case, we propose to apply association analysis, namely, *frequent itemset mining* [1] to identify the class frame patterns.

In the classical formulation of the frequent itemset mining task, the inputs are: (1) a set of *items*, and (2) a set of *transactions*, and each transaction contains a subset of items (*an itemset*). The task is to discover which sets of items (itemsets) co-occur frequently in transactions. We reuse the classical frequent itemset mining algorithms for mining *frequent axiom pattern sets*. The rationale is to discover sets of axiom patterns, which frequently appear together in the same ontology class. Such patterns would constitute a class frame pattern.

The process for mining frequent class frame patterns has three phases and is illustrated in Figure 11.

In the first phase, we mine frequent axiom patterns over an ontology $\mathcal{O}$ using the method described in Sections 4.2–4.5.

In the second phase, we apply *propositionalisation* to the mined axiom patterns. Propositionalisation is a process that transforms a structured dataset into an attribute-value (i.e., propositional) dataset. The dataset has derived propositional features which describe the structural properties of the examples. In our case, the features are represented by frequent axiom patterns, while the examples are represented by ontology classes. In the itemset mining task, the propositionalisation produces a transaction set, where each frequent axiom pattern Q^{α} represents a transaction item, and each named class A in the ontology $\mathcal{O}$ appearing on the left-hand side of any SubClassOf or EquivalentTo axiom has an associated transaction t_A. Each transaction t_i is represented by the set of all frequent axiom patterns (items) whose right-hand side matches the SubClassOf or EquivalentTo ontology axioms having A on the left-hand side. Table 2 presents a sample, illustrative result of a propositionalisation based on the example from Table 1. In the row `Heart` there is 1 in the last column, because the first pattern matches the axioms #3 and #4 in the example. Similarly, in the rows `Kidney` and `Liver` there are 1s in the first two columns, because the second and third patterns match the axioms #8 and #10, and respectively, #9 and #11.

Table 2
A propositional (attribute-value) representation where attributes (features) are constituted by frequent axiom patterns and examples are constituted by ontology classes.

	$?lhs$ SubClassOf `hasTopology` some $?c$	$?lhs$ SubClassOf `isServedBy` some $?c$	$?lhs$ SubClassOf `hasFeature` some $?c$
`Heart`	0	0	1
`Tubular`	0	0	0
`Kidney`	1	1	0
`Liver`	1	1	0

In the third phase, we apply an off-the-shelf frequent itemset mining algorithm. The result of the algorithm will be a set of itemsets, which correspond to class frame patterns (CF_A).

We define the *support* of a class frame pattern $\mathbb{Q}_A^{CF}$ over a set of transactions D_T (where each D_T corresponds to a named class from N_C appearing on the left hand-side of any SubClassOf or EquivalentTo axiom), as:

$$\sigma_{CF}(\mathbb{Q}_A^{CF}) = \left|\{t \in D_T : \mathbb{Q}_A^{CF} \subseteq t\}\right| \qquad (4)$$

The *relative support* of a class frame pattern $\mathbb{Q}_A^{CF}$ over a set of transactions D_T is the percentage of all transactions that contain all elements of $\mathbb{Q}_A^{CF}$:

$$\sigma_{CF}^r(\mathbb{Q}_A^{CF}) = \frac{|\sigma_{CF}(\mathbb{Q}_A^{CF})|}{|D_T|} \quad (5)$$

In the example from Table 2, the class frame pattern $\mathbb{Q}_A^{CF} = \{$ $?lhs$ SubClassOf `hasTopology` some $?c$, $?lhs$ SubClassOf `isServedBy` some $?c\}$ has support 2 as it is contained in two transactions.

5. Experiments and results

5.1. Frequent axiom patterns in single ontologies

To discover frequent axiom patterns in single ontologies, we encoded each ontology into a forest using the method described in Section 4.2. We then used the original SLEUTH implementation on these forests to mine frequent induced subtrees with relative support treshold of 1%. We filtered and decoded the results using the methods described in Section 4.5.

In order to present clearly the results, we divided our ontologies into six groups, depending on the number of SubClassOf and EquivalentTo axioms (ontology *size*) that they contain: up to 100 (45 ontologies), 100–1000 axioms (148 ontologies), 1000–10,000 axioms (94 ontologies), 10,000–100,000 (30 ontologies), 100,000–250,000 (9 ontologies), and over 900,000 (5 ontologies). We started the last cluster at 900,000, rather than 1,000,000, because there are no ontologies in our dataset having between 250,000 and 900,000 axioms. Therefore, we decided that three ontologies having over 900,000 are more similar to the ontologies having at least 1,000,000, so we clustered them together.

The number, support, size and depth of mined axiom patterns are presented in Figures 12a–12d. Each figure contains a box plot for every ontology group showing: the median m (horizontal line within the box); the first and third quartile (bottom and top line of the box); lowest value above $m - 1.5 \cdot IQR$ (short–horizontal line, below the box), and highest value below $m + 1.5 \cdot IQR$ (short–horizontal line, above the box). IQR is the interquartile range represented by the height of the box, and the outliers are represented as points drawn below and above of the short lines.

In Figure 12a, we present the number of mined frequent axiom patterns for each ontology group from our dataset. In Figure 12b, we present the supports for the mined frequent axiom patterns. As we used relative support threshold of 1% the lowest value of support for a given cluster of ontologies in the figure can not be lower than 1% of the size of the smallest ontology in the cluster. The maximal value for the support is bound by the size of the largest ontology.

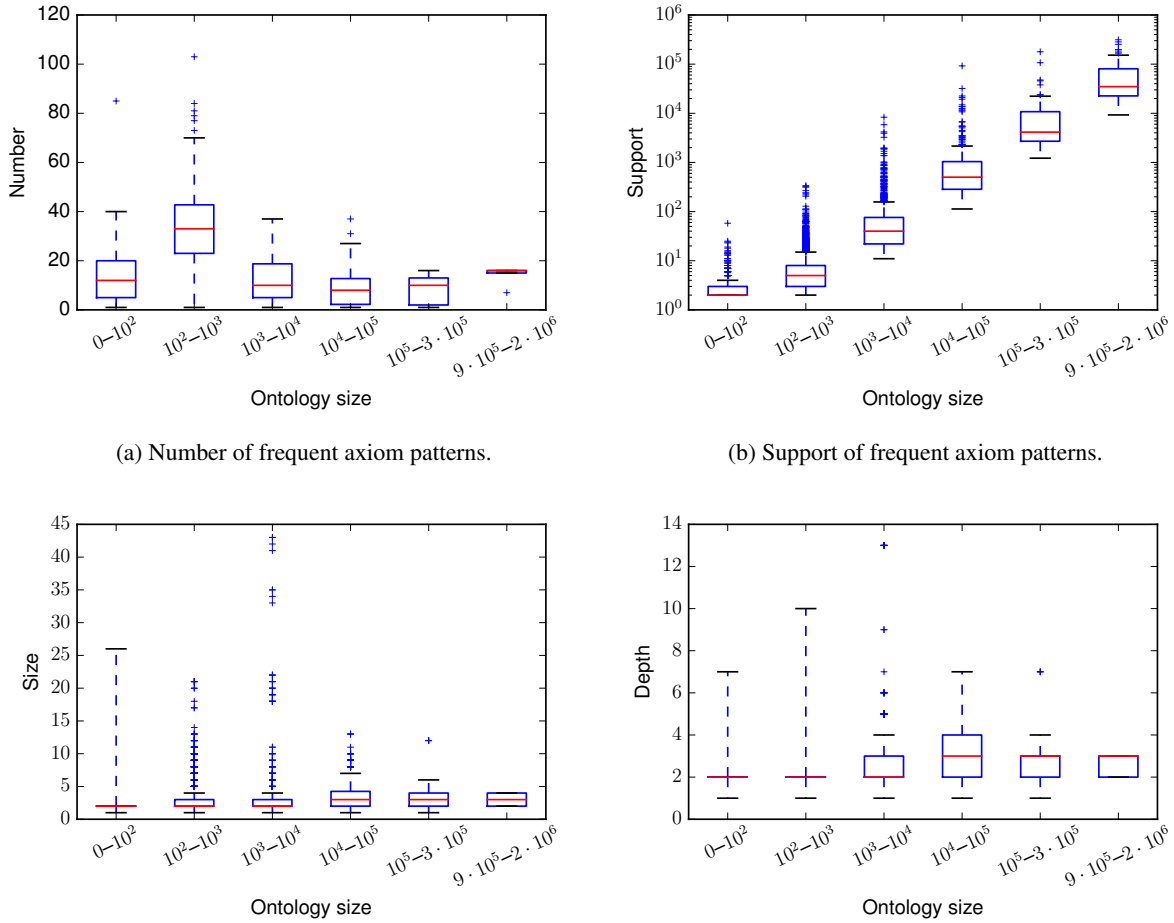
Figure 12c shows the sizes of the frequent axiom patterns that we mined. Interestingly, in larger ontologies, the median size increases, and IQR decreases. Finally, Figure 12d shows the depths of the frequent subtrees. Here, we can also observe that the median value is higher for larger ontologies.

We discovered that 96% of the ontologies (320 out of 331) in the dataset reuse patterns containing vocabulary from domain namespaces (namespaces other than `owl`, `rdf`, `rdfs` and `xsd`). For eleven of the ontologies (ATC, CMO, CRISP, FLOPO, GCO, HP, ICD10CM, ICD10PCS, ICD9CM, VT, VTO), we could not identify any patterns besides $?lhs$ SubClassOf `owl:Thing` (for GCO) and $?lhs$ SubClassOf $?c1$ (for the rest of the listed ontologies). We manually inspected these ontologies, and found that the GCO ontology contains only four classes, however, the other ten ontologies contain at least 2,000 classes. These ten ontologies have a very low average and maximum number of children, compared to the number of classes in the ontologies, which explains why there are no patterns with a named class on the left-hand side. We also noted that these ten ontologies do not contain any complex class expressions.

In the 321 ontologies, we found patterns of size at least 2; in 81 (24%) ontologies, we found patterns of size at least 5; in 17 (5%) ontologies, we found patterns of size at least 10; in 4 ontologies, patterns of size at least 20; and in one ontology (NEMO), we found 2 patterns of size 43. The biggest axiom patterns from some of the most used ontologies available in BioPortal are presented in the Appendix in Table 9.[11] The patterns with the highest support from a subset of most visited BioPortal ontologies are also presented in Table 3. The Appendix also contain breakdowns with respect to the statistics for patterns discovered in various topical categories of ontologies (Figure 18 and Figure 19). More breakdowns can be also found in the supplementary material.

One of the patterns of size 43 in the NEMO ontology (Figure 13) has resulted from repeating a substantial

[11] We display labels instead of IRIs, where possible.



(a) Number of frequent axiom patterns.

(b) Support of frequent axiom patterns.

(c) Size of frequent axiom patterns.

(d) Depth of frequent axiom patterns.

Fig. 12. Various statistics for frequent axioms patterns computed for each single ontology from our BioPortal dataset (cf. Section 4.1). The ontologies are clustered into 6 groups depending on their sizes.

fragment of the class definition in the subclasses of the class nemo:NEMO_0000093 ('scalp recorded ERP component'). By investigating this particular pattern, we found that alternative labels for this class are: 'ERP data', 'event-related potential data', and 'ERP pattern'. Thus actually, our discovered pattern represents a set of classes that represent patterns of variation in electrical activity at the scalp surface.

We discovered that 99.2% of all mined patterns contain at least one variable. Out of these, 26.1% contain a variable in the left-hand side and 89.6% contain a variable in the right-hand side.

We have also investigated the frequency statistics for namespaces occurring in axiom patterns. We looked mainly at top–level and cross–domain ontologies. We found that most patterns appear in OBO[12] ontologies, which is not surprising, given that most OBO ontologies are built using a principled approach prescribed by the OBO Foundry [31], which focuses on consistency and reuse. The `http://purl.obolibrary.org/obo/` namespace was found 3,006 times in 2,589 patterns, and 149 ontologies. This namespace is prescribed for the entities of all OBO Foundry compliant ontologies. The full table with the frequency statistics can be found in Table 4.

[12]OBO stands for Open Biomedical Ontologies. The library of OBO ontologies can be found at: `http://www.obofoundry.org/`

Table 3

The patterns with the highest support from a subset of most visited ontologies in BioPortal.

Ontology	Pattern	σ_F
PR	$?lhs$ SubClassOf: ('only_in_taxon' some 'Homo sapiens')	37854
ORDO	$?lhs$ SubClassOf: ('part_of' some $?classexpr$)	12519
NCIT	$?lhs$ SubClassOf: ('Chemotherapy_Regimen_Has_Component' some $?classexpr$)	10817
UBERON	$?lhs$ SubClassOf: ('part of' some $?classexpr$)	10716
GO	$?lhs$ SubClassOf: ('part of' some $?classexpr$)	6762
ZFA	$?lhs$ SubClassOf: ('end stage' some 'Adult')	2131
MA	$?lhs$ SubClassOf: ('part of' some $?classexpr$)	1975
GALEN	`galen:NAMEDActiveDrugIngredient` SubClassOf: $?classexpr$	1492
EDAM	`oboInOwl:ObsoleteClass` SubClassOf: $?classexpr$	904
RADLEX	`<http://www.owl-ontologies.com/Ontology1415135201.owl#RID29023>` SubClassOf: $?classexpr$	712
OBI	$?lhs$ SubClassOf: ('is quality measured as' some $?classexpr$)	266
NIFCELL	'Neuron' SubClassOf: $?classexpr$	206
PATO	$?lhs$ EquivalentTo: (('increased_in_magnitude_relative_to' some 'normal') and $?classexpr$)	100
AERO	'clinical finding' SubClassOf: $?classexpr$	50
NIFDYS	'Nervous system disease' SubClassOf: $?classexpr$	17
NIFSUBCELL	'Cellular Inclusion' SubClassOf: $?classexpr$	16

$?lhs$ equivalentTo: (($?op$ some $?classexpr1$) and (('proper_part_of' some (('is about' some (('occurs_in_response_to' some (('has_object' some (('has_role' some 'stimulus_role') and $?classexpr2$)) and 'onset_stimulus_presentation')) and 'scalp_recorded_ERP')) and 'averaged_EEG_data_set')) and ('has_proper_part' some (('is_quality_measurement_of' some (('inheres_in' some (('unfolds_in' some $?classexpr3$) and 'scalp_recorded_ERP')) and 'intensity')) and ('has_numeric_value' some `xsd:decimal`[>= "−.4"^^`xsd:decimal`]) and 'intensity_measurement_datum'))) and 'scalp_recorded_ERP_component')

Fig. 13. One of the two the longest patterns discovered in a single ontology. Both longest patterns were discovered in the NEMO ontology, and have a size of 43. The presented pattern has support 27 and depth 13. The other longest pattern is nearly identical to the presented one.

Table 4

Frequency statistics for namespaces corresponding to upper-level and cross-domain ontologies. The second column is the number of times a given namespace occurred in all patterns (multiple occurrences in a single pattern are possible). The third column shows the number of patterns containing the namespace. The fourth column shows the number of ontologies that contain at least one of these patterns.

Namespace	Overall frequency	Number of patterns	Number of ontologies
`http://purl.obolibrary.org/obo/`	3006	2589	149
`http://www.obofoundry.org/ro/ro.owl#`	85	73	19
`http://www.ifomis.org/bfo/1.1/snap#`	37	37	15
`http://www.ifomis.org/bfo/1.1/span#`	14	14	8
`http://www.ifomis.org/bfo/1.1#`	2	2	2
`http://purl.obolibrary.org/obo/bspo#`	41	29	1
`http://purl.obolibrary.org/obo/CARO#`	1	1	1

5.2. Frequent axiom patterns in a set of ontologies

To mine frequent patterns in the set of ontologies from the BioPortal repository, we used FF-SLEUTH with the support measure based on a set of forests (Section 4.4). Precisely, every axiom was translated to a tree (Section 4.2), and every ontology formed a single forest. In this way we were able to discover frequent patterns that occur in multiple ontologies independent of the relative sizes of the ontologies. If we were to combine all axioms to form a single, huge forest, then patterns from large ontologies (such as, NCIT) would dominate the results.

An experiment with a minimal support of 4 forests (i.e., 1% of 331 non-empty forests) took about 89 hours of wall-time (around 1,700 hours of CPU time) on a 2-CPU (16 threads each) server, and required

roughly 110GB of RAM. We discovered 1,935,735 frequent subtrees, out of which 640,075 (33%) were maximal—i.e., none of their supertrees were frequent. The size of the patterns (number of nodes) varies from 2 to 12, and their support is up to 29 forests (ontologies). We present the dependencies between support, size and depth in Figures 14a and 14b.

Table 5 shows the top–identified patterns with the biggest support. The top pattern with support 29 turned out to be an artifact of the way the OWL API converts OBO to OWL. We found one pattern without variables and three patterns with variables that appear in 27 ontologies.

The biggest subtree, which occurs in 14 ontologies, represents a pattern without variables (or fragment), and it is shown in Table 6. This pattern represents the logical definition of the class `'curation status specification'` (`obo:IAO_0000078`), which is used by 14 ontologies for curation purposes.

We have also examined axiom patterns that have the largest size. The sorted list is available in Table 10 in the Appendix. The size of the patterns is presented in the first column of the table and their total number of occurrences (the number of ontologies where they occur) is shown in the second column. We can observe that the largest patterns come from OBO ontologies and that they include entities from upper-level ontologies (BFO, OBI, IAO, snap, span).

Similar to our investigation for single ontologies, we have also examined which namespaces occur more frequently in ontology patterns in the entire set with a minimal support of 4. As in the previous cases, the most frequently–occurring namespaces are the ones from OBO. The full list is shown in Table 7.

5.3. Class frame patterns

We used the discovered axiom patterns to analyse class frame patterns (CFPs) as described in Section 3.2. To conduct the analysis, we used Orange3-Associate[13] to mine maximal frequent itemsets using 4 as the minimum support σ_{CF} threshold.

We have computed transactions for all ontologies and all their classes matching at least one frequent axiom pattern. Altogether, we have discovered 5,397 frequent CFPs. 2,335 (43%) of these patterns are composed of more than one axiom pattern. On average, there are 16.3 CFPs per ontology (with median value 11.0), containing an average number of axiom patterns equal to 2.7 (with median value 1.0), and with average support 233.8 (with median value 6.0). The biggest CFP (in terms of the number of axiom patterns) that we have discovered is composed of 17 axiom patterns, and the most frequent CFP that we have discovered has the support of 178,320.

In the following paragraphs, we discuss some sample CFPs from specific ontologies. The CFPs are documented in Table 8.

The *Uber Anatomy Ontology (UBERON)* [24] is a multi–species anatomy ontology that represents anatomical structures. In Table 8, we present a pattern discovered for 'mesoderm-derived structure'. Besides this pattern, we have also discovered other patterns that represent particular anatomical structures, in particular 'ectoderm-derived structure' and 'structure with developmental contribution from neural crest'.

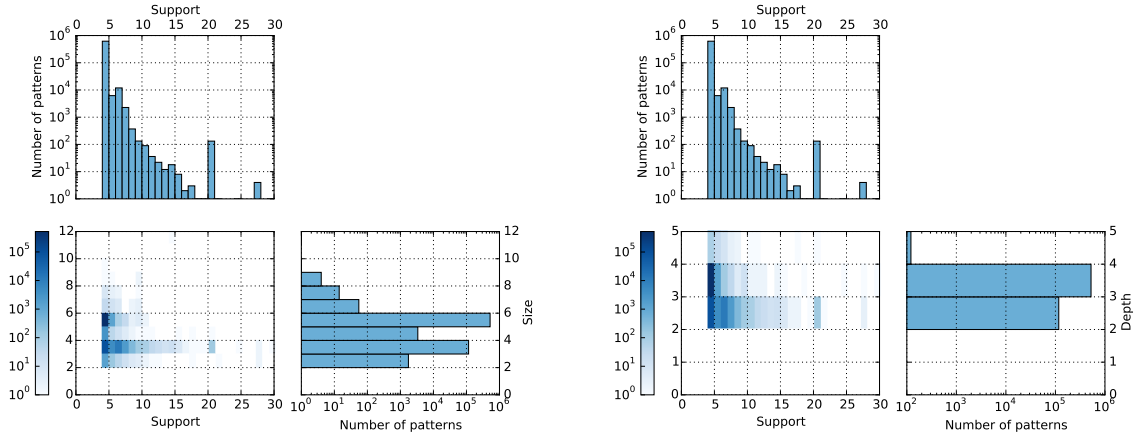
The *Ontology of Core Data Mining Entities (OntoDM)* [26] represents data mining tasks, generalizations, data mining algorithms, and more. The pattern presented in Table 8 describes the common features of subclasses of a class that only has a numeric identifier in the ontology, but no textual label. The class has nine asserted subclasses, and it is a subclass of the OBI class 'planned process'. The pattern appears in five out of these subclasses and, interestingly, it is the biggest CFP discovered for this ontology.

The *Cell Cycle Ontology (CCO)* [2] is an ontology used for representing cell cycle processes. The main entities in CCO are proteins, genes, and protein–protein interactions. Antezana et al. [2] discusses an example of the local neighborhood of the protein SWI4_YEAST using relationships (i.e., object properties) defined in CCO. The example uses relationships such as 'participates_in', 'derives_from', 'located_in' or 'transforms_into'. However, the pattern we have mined does not contain the above–mentioned relations. The mined pattern matches 561 classes out of 260,360 ones that match any frequent axiom pattern. The pattern we have discovered might represent an emerging design pattern, which was not documented in [2].

The *VIVO ontology*[14] represents researchers in the context of their experience, outputs, interests, accomplishments, and associated institutions, as well as networks of researchers. We selected this pattern (Table 8) because it shows an example of a datatype construct

[13] http://orange.biolab.si/download/

[14] http://www.vivoweb.org



(a) Distribution of sizes and supports.

(b) Distribution of depths and supports.

Fig. 14. Distributions of various statistics for patterns mined with minimal support 4. Top and right charts present histograms for one dimension each, while the charts in the middle present 2D-histograms for both statistics combined using varying color intensity.

Table 5

Top ten axiom patterns found in the set of BioPortal ontologies, sorted by descending support and size.

$\sigma_{\mathcal{F}}$	Size	Axiom pattern
29	3	?lhs SubClassOf `obo:TEMP#part_of` some ?$classexpr$
27	2	?lhs SubClassOf `snap:Role` some ?$classexpr$
27	3	`obo:IAO_0000027` SubClassOf `obo:IAO_0000030`
		('data item' SubClassOf 'information content entity')
27	3	?lhs SubClassOf ?op only (?$classexpr1$ or ?$classexpr2$ ...)
27	2	?lhs SubClassOf ?op value ?$classexpr$
26	2	?lhs SubClassOf (?op exactly 1 ?$class$)
21	2	?lhs SubClassOf `snap:Quality`
20	3	`sty:T110` SubClassOf `sty:T119`
		('Steroid' SubClassOf 'Lipid')
20	3	`sty:T028` SubClassOf `sty:T021`
		('Gene or Genome' SubClassOf 'Fully Formed Anatomical Structure')
20	3	`sty:T060` SubClassOf `sty:T058`
		('Diagnostic Procedure' SubClassOf 'Health Care Activity')

Table 6

An axiom pattern without variables (or fragment) corresponding to the biggest subtree discovered in a set of ontologies. It has size 12, depth 3 and support 14.

'curation status specification' EquivalentTo {'uncurated', 'to be replaced with external ontology term', 'pending final vetting', 'ready for release', 'metadata incomplete', 'requires discussion', 'metadata complete', 'organizational term', 'example to be eventually removed'}

in an axiom pattern, which is not present in any of the other presented patterns.

The *Protein Ontology (PR)* [25] represents protein-related entities. This CFP has a large support of 18,207, while having an above the average number of frequent axiom patterns.

The *Clusters of Orthologous Groups (COG) Analysis Ontology (CAO)* [20] is designed for supporting the COG enrichment study. The selected CFP contains cardinality restrictions, which are a rare occurrence in other mined patterns.

The *GALEN ontology* [29] represents concepts related to anatomy, drugs, diseases, signs and symptoms.

Table 7
Namespaces occurring in frequent patterns mined with minimal support 4.

Namespace	Overall frequency	Number of patterns
http://purl.obolibrary.org/obo/	1,794,328	639,676
http://purl.obolibrary.org/obo/SSB#	1,555	1,555
http://edamontology.org/	461	243
http://purl.bioontology.org/ontology/STY/	262	131
http://www.ifomis.org/bfo/1.1/snap#	76	39
http://www.ifomis.org/bfo/1.1/span#	58	26
http://www.obofoundry.org/ro/ro.owl#	12	12
http://www.geneontology.org/formats/oboInOwl#	8	8
http://www.ifomis.org/bfo/1.1#	5	5
http://purl.org/obo/owl/GO#	4	3
http://purl.org/biotop/biotop.owl#	3	3
http://purl.org/obo/owl/PATO#	3	3
http://purl.obolibrary.org/obo/TEMP#	2	2
http://purl.obolibrary.org/obo/OBO_REL#	1	1
http://purl.org/obo/owl/OBO_REL#	1	1

The presented CFP shows an example that is composed of complex axiom patterns.

The *Ontology for Biomedical Investigations (OBI)* [6] has resulted from a cross-community effort to provide a resource that represents biomedical investigations to facilitate interpretation of the experimental process. We decided to present this CFP because it contains a named class on the left-hand side.

5.4. Mining documented class frame patterns

One of the research questions that we are trying to answer is whether our methods are able to mine axiom patterns of ODPs described in literature. Figure 15 shows the CFPs that we have automatically mined for the Cell Line Ontology (CLO). After a manual investigation, we have subsequently established that the mined patterns reflect the 'Cell Line Cells' design pattern proposed by Sarntivijai et al. [30]. CLO is one of the largest BioPortal ontologies from our dataset (it contains 114,843 SubClassOf and EquivalentTo axioms). Some of the axiom patterns that we have mined also have a high–absolute support, reaching the value of 21,698. The CFPs shown in Figure 15 have sizes between 2 and 4, and the support value ranging from 9 to 728.

One surprising finding is the fact that we discovered several CFPs, which contain a part that is not included in the original ODP, namely: ('has quality at some time' some 'male') (or ('has quality at some time' some 'female')). This part is depicted in Figure 15 with a dashed line. This finding may indicate a concept drift. In order to answer whether this part is a plausible addition to the ODP, we would need to run another study. Another finding that validates the mined patterns comes from Sarntivijai et al. [30]. The paper describes the addition of 1,622 new cell lines from the Japan RIKEN Cell Bank to CLO, which is evidenced in our discovered frequent axiom pattern: SubClassOf 'is in cell line repository' value 'RIKEN Cell Bank', with an absolute support of 1,622.

The Manchester ODP Catalog and Ontorat [41] are two pattern repositories for biomedical ontologies that document or refer to patterns from ontologies contained in our dataset: CLO, OBI, OOEVV, BCGO, and CCO. We investigated whether we were able to mine the documented patterns. Our manual inspection confirmed that we were able to mine the patterns in CLO (described above), as well as the 'Assay' pattern from OBI and an instantiation of a fragment of the 'Device' pattern from the same ontology, and a pattern concerning the main classes of OOEVV [7].

6. Discussion

6.1. Research questions

1. Do certain patterns recur in ontologies? Can we generalize over such patterns to mine more generic templates?

We found patterns in every ontology in the experiment, with the exception of those that did not have any SubClassOf or EquivalentTo axioms (Section 4.1).

Table 8

Selected class frame patterns (CFP). First column displays the name of the ontology where the CFP was found. Second column contains the relative support σ^r_{CF} and the support σ_{CF} values (in parantheses). Third column shows the CFP. E.g., the UBERON ontology contains a CFP composed of four frequent axiom patterns. The variables on the right-hand side of the axiom patterns (?c, ?p, etc.) have been renamed to reflect that they are local in scope to the each axiom pattern, and thus they may bind to different entities within the scope of a class frame.

Ontology	σ^r_{CF} (σ_{CF})	Class frame pattern (CFP)
UBERON	0.08% (8)	?lhs SubClassOf: 'mesoderm-derived structure' ?lhs SubClassOf: ('part_of' some ?$c1$) ?lhs SubClassOf: ('develops_from' some ?$c2$) ?lhs SubClassOf: ('contributes to morphology of' some ?$c3$)
OntoDM	1.85% (5)	?lhs SubClassOf: <http://kt.ijs.si/panovp/OntoDM#OntoDM_000290> ?lhs SubClassOf: (?$p1$ some ('ensemble of generalizations' or 'single generalization')) ?lhs SubClassOf: ('has_specified_output' some (?$classexpr1$ or ?$classexpr2$)) ?lhs SubClassOf: ('has_specified_input' some ?$c1$) ?lhs SubClassOf: (?$p2$ some 'DM-dataset') ?lhs SubClassOf: ('realizes' some ('is_concretization_of' some ?$c2$))
CCO	0.21% (561)	?lhs SubClassOf: 'protein' ?lhs SubClassOf: ('enables' some ?$c1$) ?lhs SubClassOf: ('inheres in' some 'Homo sapiens') ?lhs SubClassOf: ('part of' some ?$c2$) ?lhs SubClassOf: ('involved in' some ?$c3$) ?lhs SubClassOf: ('is orthologous to' some ?$c4$) ?lhs SubClassOf: ('bearer of' some ?$c5$) ?lhs SubClassOf: ('is paralogous to' some ?$c6$)
VIVO	4.38% (5)	?lhs SubClassOf: ('date/time interval' only 'Date/Time Interval') ?lhs SubClassOf: ('description' only rdfs:Literal)
PR	23.90% (18,207)	?lhs SubClassOf: 'Homo sapiens protein' ?lhs SubClassOf: ('only_in_taxon' some 'Homo sapiens') ?lhs SubClassOf: ('has_gene_template' some ?$c1$) ?lhs EquivalentTo: (('only_in_taxon' some 'Homo sapiens') and ?$classexpr1$)
CAO	15.09% (24)	?lhs SubClassOf: 'COG category protein' ?lhs SubClassOf: ('is_member_of' some ?$c1$) ?lhs EquivalentTo: ('COG category protein' and ('denoted_by' min 1 ?$c2$))
GALEN	0.29% (26)	?lhs EquivalentTo: ((?$p1$ some ((?$p2$ some ((?$p3$ some ?$c1$) and ?$classexpr1$)) and ?$classexpr2$)) and ?$classexpr3$) ?lhs EquivalentTo: ((?$p4$ some ((?$p5$ some ?$c2$) and ?$classexpr4$)) and (?$p6$ some ?$c3$)) ?lhs EquivalentTo: ((?$p7$ some ?$c5$) and galen:BodyStructure)
OBI	0.39% (5)	'assay' SubClassOf: ?$classexpr1$?lhs SubClassOf: ('has_specified_output' some (('is about' some ?$c1$) and ?$classexpr2$)) ?lhs SubClassOf: ('has_specified_input' some ?$c2$) ?lhs SubClassOf: ('achieves_planned_objective' some ?$c3$)

In 320 out of 331 ontologies (97%) in the dataset, we found patterns containing vocabulary from domain namespaces (i.e., namespaces other than `owl`, `rdf`, `rdfs` and `xsd`).

We also noted that most patterns contain vocabulary from OBO ontologies (Table 4). This finding hints at the fact that modeling patterns and reuse are more prevalent in OBO ontologies than in the other ontologies in the dataset. This fact is not surprising as OBO ontologies follow the principles set forth by the OBO Foundry [31], which prescribe a strict set of rules for reuse and orthogonality of ontologies.

We have also observed that the median fragment size for smaller and larger ontologies (with less or more than 1,000 axioms, respectively) is fairly similar, between 2 and 3 (see Figure 12c), although there are variations in IQR. This finding may indicate that most patterns are still fairly simple, rather than com-

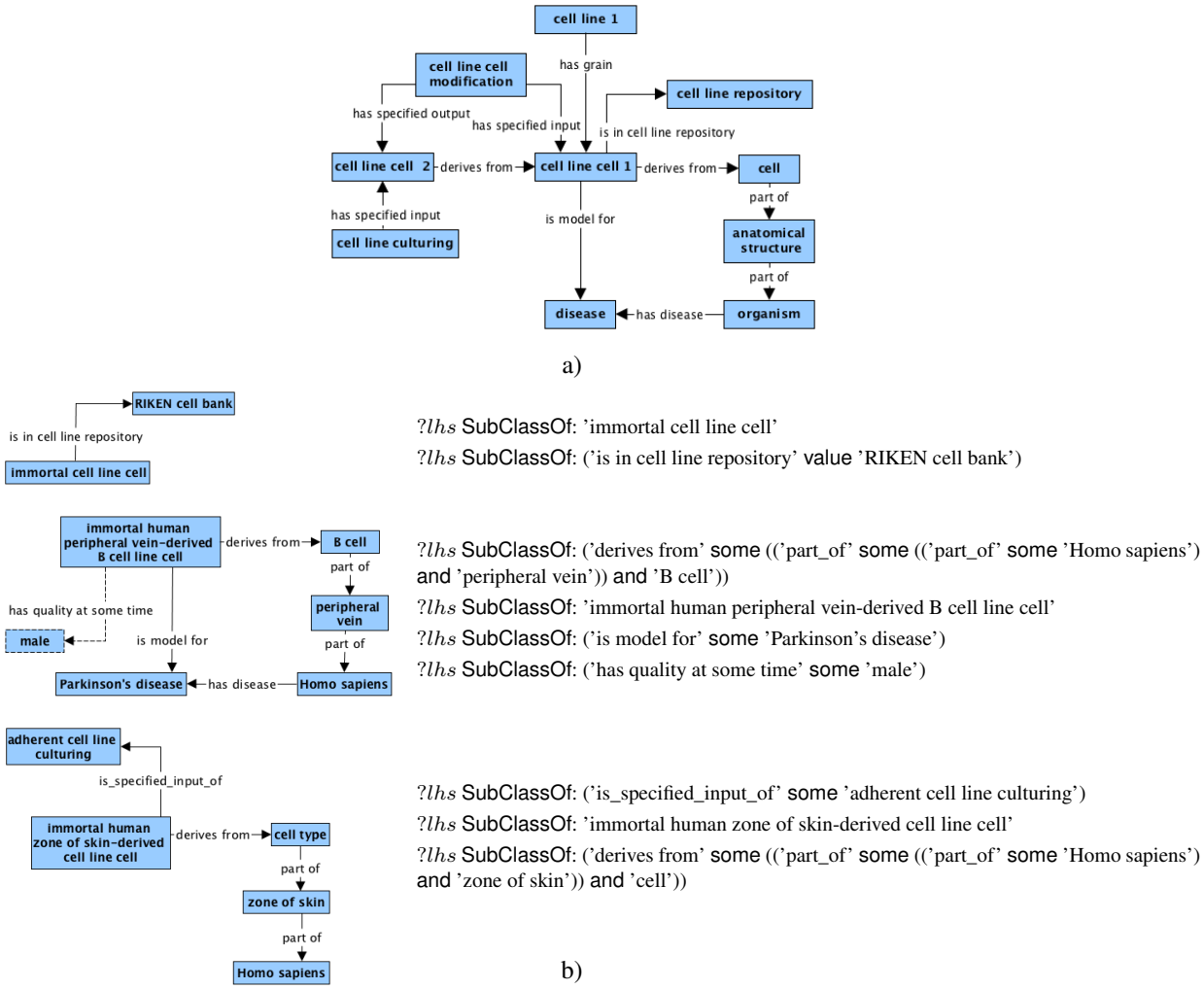


Fig. 15. a) 'Cell Line Cells' design pattern for the CLO ontology [30] (top). b) The selected corresponding class–frame fragments, which we have automatically mined (middle and bottom part of the figure).

plex expressions, and are usually of the size of two or three. We discovered that the majority (99.2%) of all mined axiom patterns contain at least one variable, out of which 89.6% contain a variable in the right-hand side of the axiom.

2. Do such patterns appear within a group of ontologies?

We found that ontology patterns exist, not only in single ontologies, but across the set of investigated ontologies. In the latter case, the longest patterns discovered from the set of all ontologies (Table 10), are patterns without variables. They represent fragments from OBO ontologies, which have likely been copied from other ontologies. For example, the 'curation status specification' class (Table 6) is originally defined in the file `ontology-metadata.owl`,[15] but is copied in fourteen of the ontologies in our dataset. This finding hints that these 14 ontologies may have used the MIREOT principles [8] to copy just parts of a source ontology into the target ontology. MIREOT defines the minimum information needed to reference external ontologies, and many OBO ontologies use it. The finding also suggests that the 14 ontologies have been built using a similar development process (e.g., they all use the same curation statuses). This kind of similarity in the

[15] `http://information-artifact-ontology.googlecode.com/svn/trunk/src/ontology/ontology-metadata.owl`

development processes is expected in a focused community, such as the OBO one.

We also noted that several of the rows in Table 10 are fragments (i.e., APNV) of upper ontologies—such as BFO—or cross–domain ontologies—such as, OBI or IAO. One question that arises is whether these fragments may represent reusable ontology modules [32], which would be valuable also outside of the OBO community. To facilitate their reuse, such modules could be made available separately from the ontologies from which they originate.

3. Do such patterns exist on the axiom level? Do they exist on the level of sets of axioms?

We found patterns on both levels. We were able to mine frequent patterns from every ontology that contained SubClassOf or EquivalentTo axioms. In Section 5.3, we presented a subset of the frequent class frame patterns (CFP) that we mined. We have found 2,335 CFPs composed of more than one frequent axiom pattern, with an average of 16.30 CFPs per ontology.

This result is intriguing taking into account that Mortensen et al. [23] found modest reuse of ODPs in BioPortal ontologies. These results are, however, not contradictory. The approach taken by Mortensen et al. is top-down—they test the occurrence of a set of several predefined patterns in the ontology dataset. This study found that the ontologies in BioPortal contain some of the structural patterns from Manchester ODP Catalog, and a few high–level content patterns from the ODP Portal.

In contrast, our approach mines patterns bottom–up, and can also detect parts of specific content ODPs. We call our mined patterns "emerging", as they may not comply to predefined ODPs in existing repositories. Yet, these patterns appear in the studied ontologies, likely because they are valuable to the ontology authors and users.

4. Are we able to automatically detect fragments of documented ODPs?

We were able to establish manually that some of the patterns that we have automatically mined are fragments of ODPs proposed in literature. We have detected fragments of ODPs for CLO, OBI (the 'Assay' pattern), and OOEVV, which were documented in the Manchester ODP Catalog and in the Ontorat repository. For the 'Device' pattern for OBI from Ontorat, we have been able to detect an instantiation of a fragment of the pattern ($?lhs$ SubClassOf: 'has function' some 'measure function'), and a more generic pattern fitting its part ({$?lhs$ SubClassOf: (($?p$ some $?c$) and ?), $?lhs$ SubClassOf: ('has part' some $?c$)}), but we were unable to mine its part which uses the property 'is_manufactured_by' since the frequency of the usage of this property in the ontology axioms was below our support threshold. Our algorithm did not mine fragments of the documented pattern for CCO. By manual inspection we have found that some properties documented in the paper are not appearing in the ontology file we mined, e.g. we could not find the property 'participates_in' nor 'located_in'. Nevertheless, we mined another pattern for CCO (Table 8), which might represent an emerging design pattern. We also did not mine the BCGO pattern (adding new mouse strains with annotations using IAO properties) since this pattern concerns annotations, not logical axioms. However, we were able to mine different patterns in the BCGO ontology.

Besides the exact parts of the proposed ODPs, we also found two other types of constructs:

1. *Frequent patterns that are more specific than parts of the proposed ODP*. The more specific patterns are exemplified in Figure 15. The mined patterns show examples of a 'cell line cell', 'cell', 'anatomical structure', 'organism', 'disease', 'cell line repository', and 'cell line modification', which are frequently appearing in the CLO ontology. The patterns describe particular types of cell line cells (e.g., immortal human zone of skin-derived cell line cell), cells (e.g., B cell), anatomical structures (e.g., zone of skin), etc., or even a cell line repository (RIKEN cell bank).
2. *A drift or a novelty*. We found that many class frame patterns mined in the CLO ontology contain a part that is not included in the original ODP ('has quality at some time' some 'male', shown in Figure 15).

We note that BioPortal hosts a relatively well–described set of ontologies. The ontologies are documented either in scientific publications, and/or on the webpages of the projects that developed them. Thus, it allowed us to identify the patterns used in the ontologies' construction, and then to check whether our algorithm can mine the documented patterns. We also note that our approach is generic and it can be applied in other domains and with other datasets.

6.2. Supported pattern types

The authors of [4,12] distinguish six types of ontology design patterns: content, structural, correspondence, reasoning, presentation, and lexico-syntactic. Our method is suited to mine three of the six types of patterns: content, some structural (logical) and some correspondence (alignment) ODPs. However, our approach is not suited for discovering reasoning, presentation and lexico-syntactic ODPs.

The content ODPs are the main target of our method. We have shown in Section 5.4 that we can automatically mine a part of the 'Cell Line Cells' content ODP. We have also mined frequent CFPs which contain specific domain vocabulary (Table 8). We have also shown that we can mine patterns that contain mostly variables—corresponding to a subtype of structural ODPs, namely logical ODPs. For example see Figure 16.

There are two types of correspondence ODPs: re-engineering and alignment ODPs. Our method cannot mine re-engineering patterns, which represent transformation rules to create a new ontology from elements of a source model. However, our method can mine some of the alignment ODPs, in particular, those that express class equivalence and class subsumption. Our method can detect if an ontology reuses parts of another ontology, which comes from a different namespace. For example see Figure 17.

6.3. Possible reasons for pattern occurrence

Throughout the paper, we mentioned possible reasons for which patterns occur in ontologies, summarized as follows:

- *Copying a fragment from an ontology* — exemplified by the 'curation status specification' pattern in Table 6. This case likely occurs when developers in a community reuse a generic ontology part that acts like an ontology module.
- *Repeating a substantial fragment of the class definition in the subclasses of the class* —exemplified by the pattern found in the subclasses of 'scalp recorded ERP component' class from the NEMO ontology (Figure 13). This case likely occurs because of implicit or explicit patterns that occur in the development of specific ontologies. In some cases, such patterns are enforced by the user interface, e.g., through the use of templates [38].
- *Reusing documented and recommended ontology design patterns* — exemplified by the mined fragments of 'Assay' ODP (Table 8). This case likely occurs because ontology developers have made an explicit effort to either (1) reuse an existing ODP, or (2) document after-the-fact a useful ODP that emerged from their development in a scientific publication or in an ontology repository.

6.4. Possible uses

We envision several uses of the methods and findings in this paper. First, our approach can be used to extract frequent fragments (APNV) from sets of ontologies—like the one shown in Table 6. These fragments may form generic reusable modules that might benefit the development of other ontologies. Second, ontology authors may run the mining algorithm to discover implicit patterns in ontologies that are developed collaboratively, and potentially adopt some of these patterns as recommended practices. Third, the mined patterns may be inspected manually, and then submitted to one of the online pattern repositories to enable their reuse. And fourth, the mined patterns can be used to create custom user interfaces—for example, in the form of templates—to enable their easier authoring and error checking. For instance, a custom user interface may allow only the entry of constructs that are conforming to the pattern definition, and thus, possibly, reducing authoring errors.

6.5. Our approach versus RIO

The RIO method developed by Mikroyannidi et al. [21,22] computes clusters of ontology entities. Then, for each cluster, it computes a set of axiom generalizations. Each generalization has an associated set of one or more matching axioms, which contain an entity from the cluster. A cluster aggregates axiom generalizations, which describe similar usages of subsets of clustered entities in the axioms. In contrast to our approach, generalizations within one cluster may involve largely disjoint sets of clustered entities. An entity may also appear in an axiom in various positions, both on the left-hand side and the right-hand side of the axiom.

We use the VIVO ontology to exemplify the differences between our approach and RIO's. One of the clusters generated by RIO for the VIVO ontology gathers 9 entities. Five of these entities also match our class frame pattern (CFP) shown in Table 8. The RIO clus-

$?lhs$ EquivalentTo: ($?op1$ some $?classexpr1$) and ($?op2$ some $?classexpr2$) and ($?op3$ some $?classexpr3$)

Fig. 16. A sample logical ODP found in *National Cancer Institute Thesaurus* (NCIT). The ODP consists only of variables and OWL vocabulary.

$?lhs$ EquivalentTo: `snap:MaterialEntity` and (<`http://purl.org/obo/owl/OBO_REL#bearer_of`> some (`obo:OBI_0000294` some $?classexpr$))

Fig. 17. A sample alignment ODP found in *Ontology for Drug Discovery Investigations* (DDI). The ODP uses vocabulary from three different namespaces, while being present in an ontology using yet another namespace as the base.

ter is described by 43 axiom generalizations, which also include 2 axiom generalizations that correspond to the axiom patterns that we mined as a CFP for VIVO. However, the axiom generalizations in RIO only characterize subsets of the entities. Without further analysis, it is impossible to know what is the overlap between the generalizations. In this particular cluster, most axiom generalizations (more than a half of them – 22) cover only single axioms. One of the axiom generalizations from this cluster is "$?Event$ SubClassOf $?cluster_{10}$", which matches 10 axioms. However, all of the axioms match the same single entity from the cluster, namely `Event`, which appears on the right-hand side of each of these axioms. We conclude that the axiom generalizations computed by RIO cannot be combined together to form a description of the shared attributes of all the entities from a cluster in the way that our class frame patterns can describe a set of classes forming emerging design patterns.

6.6. Limitations

Our approach has several limitations. One limitation is that we can only discover patterns occurring in the ontology itself. That is, we can only discover what is frequently expressed through ontology axioms. Please note that not everything, which is expressed visually in the ODPs from literature (e.g., using UML), can be represented with the types of OWL axioms that we consider in our approach. The reason is that these OWL axioms have a tree–shaped, and variable–free form. As a consequence, our mined class–frame patterns also have a tree–shaped form. In addition, it is important to notice that the ODPs proposed in the literature are just a recommendation, and the actual ontology modeling may not entirely conform to the recommended patterns.

Another limitation is also related to the tree–shaped form of the OWL axioms, and the effect of our two–step mining process of the class frame patterns. We mine class frames on top of the already discovered frequent axiom patterns. It might be the case that the variables appearing in a class frame pattern (as part of different axiom patterns) refer to the same entity. However, we cannot say currently whether this is the case. We can also not mine cyclic patterns. The motivation for our two-step method is to make the mining of class frames computationally feasible. Without this constraint, the search space for data mining algorithms becomes prohibitively large.

Although we are able to detect (emerging) design patterns automatically, our method cannot confirm whether a mined pattern is indeed a fragment of an ODP, and this needs to be confirmed manually.

7. Conclusions

In this paper, we described a two-step approach for automatically detecting axiom patterns in ontologies. Our approach is able to detect three different types of patterns: axiom patterns with variables, axiom patterns without variables (a.k.a, ontology fragments), and class frame patterns. We described the two methods used in our approach: (1) a tree mining method for discovering frequently recurring ontology axiom patterns; and (2) an association analysis method to discover frequent class frame patterns. We conducted an experimental analysis on a corpus of 331 BioPortal ontologies, and found that all ontologies in the corpus contain at least one of the three types of patterns. We also extracted 'emerging' design patterns (frequent class frame patterns) from the ontology corpus. We could confirm manually that some of these patterns are fragments of ODPs documented in the literature. Our approach is generic, and can be applied to ontologies from any domain.

As future work, we would like to explore application scenarios that would benefit from some form of inference, for which we would extend our approach to take such inference into account. We would also like to further apply and test our methods on other ontology repositories. We envisage that our data–driven approach for identifying ontology patterns will help expose emerging design patterns and potential ontology

modules, and it will ultimately lead to a better reuse across ontologies in all domains.

Acknowledgements. This work was partially supported by the PARENT-BRIDGE program of Foundation for Polish Science, co-financed from European Union, Regional Development Fund (Grant No POMOST/2013-7/8). Agnieszka Ławrynowicz acknowledges the support from the National Science Center (Grant No 2014/13/D/ST6/02076). This work is also supported in part by grants GM086587 and GM103316 from the US National Institutes of Health.

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994. URL http://www.vldb.org/conf/1994/P487.PDF.

[2] Erick Antezana, Mikel E Aranguren, Ward Blondé, Aitzol Illarramendi, Iñaki Bilbao, Bernard De Baets, Robert Stevens, Vladimir Mironov, and Martin Kuiper. The Cell Cycle Ontology: an application ontology for the representation and integrated analysis of the cell cycle process. *Genome Biology*, 10(5):R58, 2009. DOI https://doi.org/10.1186/gb-2009-10-5-r58.

[3] Mikel E Aranguren, Erick Antezana, Martin Kuiper, and Robert Stevens. Ontology design patterns for bio-ontologies: a case study on the Cell Cycle Ontology. *BMC Bioinformatics*, 9(5):S1, 2008. DOI https://doi.org/10.1186/1471-2105-9-S5-S1.

[4] Eva Blomqvist and Kurt Sandkuhl. Patterns in ontology engineering: Classification of ontology patterns. In Chin-Sheng Chen, Joaquim Filipe, Isabel Seruca, and José Cordeiro, editors, *ICEIS 2005, Proceedings of the Seventh International Conference on Enterprise Information Systems, Miami, USA, May 25-28, 2005*, pages 413–416, 2005.

[5] Eva Blomqvist, Pascal Hitzler, Krzysztof Janowicz, Adila Krisnadhi, Tom Narock, and Monika Solanki. Considerations regarding ontology design patterns. *Semantic Web*, 7(1):1–7, 2016. DOI https://doi.org/10.3233/SW-150202.

[6] Ryan R Brinkman, Mélanie Courtot, Dirk Derom, Jennifer M Fostel, Yongqun He, Phillip Lord, James Malone, Helen Parkinson, Bjoern Peters, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Larisa N Soldatova, Jr. Stoeckert, Christian J, Jessica A Turner, and Jie Zheng. Modeling biomedical experimental processes with OBI. *Journal of Biomedical Semantics*, 1(Suppl 1):S7, 2010. DOI https://doi.org/10.1186/2041-1480-1-S1-S7.

[7] Gully A. P. C. Burns and Jessica A. Turner. Modeling functional magnetic resonance imaging (fMRI) experimental variables in the ontology of experimental variables and values (OoEVV). *NeuroImage*, 82:662–670, 2013. DOI https://doi.org/10.1016/j.neuroimage.2013.05.024.

[8] Melanie Courtot, Frank Gibson, Allyson L. Lister, James Malone, Daniel Schober, Ryan R. Brinkman, and Alan Ruttenberg. MIREOT: the minimum information to reference an external ontology term. *Applied Ontology*, 6(1):23–33, 2011. DOI https://doi.org/10.3233/AO-2011-0087.

[9] Nick Drummond, Alan L. Rector, Robert Stevens, Georgina Moulton, Matthew Horridge, Hai Wang, and Julian Seidenberg. Putting OWL in order: Patterns for sequences in OWL. In Bernardo Cuenca Grau, Pascal Hitzler, Conor Shankey, and Evan Wallace, editors, *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006. URL http://ceur-ws.org/Vol-216/submission_12.pdf.

[10] Mikel Egana, Alan Rector, Robert Stevens, and Erick Antezana. Applying ontology design patterns in bio-ontologies. In Aldo Gangemi and Jérôme Euzenat, editors, *Knowledge Engineering: Practice and Patterns*, volume 5268 of *Lecture Notes in Computer Science*, pages 7–16. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-87695-3. DOI https://doi.org/10.1007/978-3-540-87696-0_4.

[11] Aldo Gangemi. Ontology design patterns for semantic web content. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*, volume 3729 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2005. DOI https://doi.org/10.1007/11574620_21.

[12] Aldo Gangemi and Valentina Presutti. Ontology design patterns. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 221–243. Springer, 2009. DOI https://doi.org/10.1007/978-3-540-92673-3_10.

[13] Peter Haase, Holger Lewen, Rudi Studer, Duc Thanh Tran, Michael Erdmann, Mathieu d'Aquin, and Enrico Motta. The NeOn ontology engineering toolkit. In *WWW 2008 Developers Track*, April 2008.

[14] Matthew Horridge and Sean Bechhofer. The OWL API: A java API for OWL ontologies. *Semantic Web*, 2(1):11–21, 2011. DOI https://doi.org/10.3233/SW-2011-0025.

[15] Matthew Horridge and Peter Patel-Schneider. OWL 2 Web Ontology Language Manchester syntax (second edition). W3C note, W3C, December 2012. http://www.w3.org/TR/2012/NOTE-owl2-manchester-syntax-20121211/.

[16] Matthew Horridge, Tania Tudorache, Jennifer Vendetti, Csongor I. Nyulas, Mark A. Musen, and Natalya F. Noy. Simplified OWL ontology editing for the web: Is WebProtégé enough? In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web – ISWC 2013*, volume 8218 of *Lecture Notes in Computer Science*, pages 200–215. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-41334-6. DOI https://doi.org/10.1007/978-3-642-41335-3_13.

[17] Muhammad Tahir Khan and Eva Blomqvist. Ontology Design Pattern Detection - Initial Method and Usage Scenarios. In *Proceedings of the 4th International Conference on Advances in Semantic Processing, SEMAPRO*, pages 19 – 24, Florence, Italy, 2010. http://www.thinkmind.org/index.php?view=article&articleid=semapro_2010_1_40_50071.

[18] Holger Knublauch, Matthew Horridge, Mark A. Musen, Alan L. Rector, Robert Stevens, Nick Drummond, Phillip W. Lord, Natalya Fridman Noy, Julian Seidenberg, and Hai Wang. The Protégé OWL experience. In Bernardo Cuenca Grau, Ian Horrocks, Bijan Parsia, and Peter F. Patel-Schneider, editors, *Proceedings of the OWLED*05 Workshop on OWL: Experiences and Directions, Galway, Ireland, November 11-12, 2005*, volume 188 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005. URL http://ceur-ws.org/Vol-188/sub14.pdf.

[19] Agnieszka Lawrynowicz and Jedrzej Potoniec. Fr-ONT: An algorithm for frequent concept mining with formal ontologies. In Marzena Kryszkiewicz, Henryk Rybinski, Andrzej Skowron, and Zbigniew W. Ras, editors, *Foundations of Intelligent Systems - 19th International Symposium, ISMIS 2011, Warsaw, Poland, June 28-30, 2011. Proceedings*, volume 6804 of *Lecture Notes in Computer Science*, pages 428–437. Springer, 2011. DOI https://doi.org/10.1007/978-3-642-21916-0_46.

[20] Yu Lin, Zuoshuang Xiang, and Yongqun He. Towards a semantic web application: Ontology-driven ortholog clustering analysis. In Olivier Bodenreider, Maryann E. Martone, and Alan Ruttenberg, editors, *Proceedings of the 2nd International Conference on Biomedical Ontology, Buffalo, NY, USA, July 26-30, 2011*, volume 833 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011. URL http://ceur-ws.org/Vol-833/paper5.pdf.

[21] Eleni Mikroyannidi, Luigi Iannone, Robert Stevens, and Alan L. Rector. Inspecting regularities in ontology design using clustering. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, volume 7031 of *Lecture Notes in Computer Science*, pages 438–453. Springer, 2011. DOI https://doi.org/10.1007/978-3-642-25073-6_28.

[22] Eleni Mikroyannidi, Robert Stevens, Luigi Iannone, and Alan L. Rector. Analysing syntactic regularities and irregularities in SNOMED-CT. *Journal of Biomedical Semantics*, 3:8, 2012. DOI https://doi.org/10.1186/2041-1480-3-8.

[23] Jonathan Mortensen, Matthew Horridge, Mark A. Musen, and Natalya Fridman Noy. Modest use of ontology design patterns in a repository of biomedical ontologies. In Eva Blomqvist, Aldo Gangemi, Karl Hammar, and Mari Carmen Suárez-Figueroa, editors, *Proceedings of the 3rd Workshop on Ontology Patterns, Boston, USA, November 12, 2012*, volume 929 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012. URL http://ceur-ws.org/Vol-929/paper4.pdf.

[24] Christopher J Mungall, Carlo Torniai, Georgios V Gkoutos, Suzanna E Lewis, and Melissa A Haendel. Uberon, an integrative multi-species anatomy ontology. *Genome Biology*, 13(1):R5, 2012. DOI https://doi.org/10.1186/gb-2012-13-1-r5.

[25] Darren A. Natale, Cecilia N. Arighi, Judith A. Blake, Carol J. Bult, Karen R. Christie, Julie Cowart, Peter D'Eustachio, Alexander D. Diehl, Harold J. Drabkin, Olivia Helfer, Hongzhan Huang, Anna Maria Masci, Jia Ren, Natalia V. Roberts, Karen Ross, Alan Ruttenberg, Veronica Shamovsky, Barry Smith, Meher Shruti Yerramalla, Jian Zhang, Aisha Al-Janahi, Irem Çelen, Cynthia Gan, Mengxi Lv, Emily Schuster-Lezell, and Cathy H. Wu. Protein ontology: a controlled structured network of protein entities. *Nucleic Acids Research*, 42(Database-Issue):415–421, 2014. DOI https://doi.org/10.1093/nar/gkt1173.

[26] Pance Panov, Larisa Soldatova, and Saso Dzeroski. Ontology of core data mining entities. *Data Mining and Knowledge Discovery*, 28(5-6):1222–1265, 2014. ISSN 1384-5810. DOI https://doi.org/10.1007/s10618-014-0363-0.

[27] Bijan Parsia, Sebastian Rudolph, Markus Krötzsch, Peter Patel-Schneider, and Pascal Hitzler. OWL 2 web ontology language primer (second edition). Technical report, W3C, December 2012. http://www.w3.org/TR/2012/REC-owl2-primer-20121211/.

[28] Valentina Presutti, Eva Blomqvist, Enrico Daga, and Aldo Gangemi. Pattern-based ontology design. In Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta, and Aldo Gangemi, editors, *Ontology Engineering in a Networked World*, pages 35–64. Springer, 2012. DOI https://doi.org/10.1007/978-3-642-24794-1_3.

[29] Alan L. Rector, Jeremy Rogers, Pieter E. Zanstra, and Egbert J. van der Haring. OpenGALEN: Open source medical terminology and tools. In *AMIA 2003, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 8-12, 2003*. AMIA, 2003. URL http://knowledge.amia.org/amia-55142-a2003a-1.616734/t-002-1.618748/f-001-1.618749/a-364-1.618993/a-365-1.618990.

[30] Sirarat Sarntivijai, Yu Lin, Zuoshuang Xiang, Terrence F. Meehan, Alexander D. Diehl, Uma D. Vempati, Stephan C. Schürer, Chao Pang, James Malone, Helen E. Parkinson, Yue Liu, Terue Takatsuki, Kaoru Saijo, Hiroshi Masuya, Yukio Nakamura, Matthew H. Brush, Melissa Haendel, Jie Zheng, Christian J. Stoeckert Jr., Bjoern Peters, Christopher J. Mungall, Thomas E. Carey, David J. States, Brian D. Athey, and Yongqun He. CLO: the cell line ontology. *Journal of Biomedical Semantics*, 5:37, 2014. DOI https://doi.org/10.1186/2041-1480-5-37.

[31] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255, 2007. DOI https://doi.org/10.1038/nbt1346.

[32] Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra, editors. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*. Springer, 2009. ISBN 978-3-642-01906-7. DOI https://doi.org/10.1007/978-3-642-01907-4.

[33] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. The NeOn methodology for ontology engineering. In Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta, and Aldo Gangemi, editors, *Ontology Engineering in a Networked World.*, pages 9–34. Springer, 2012. DOI https://doi.org/10.1007/978-3-642-24794-1_2.

[34] Ondrej Sváb-Zamazal and Vojtech Svátek. Analysing ontological structures through name pattern tracking. In Aldo

Gangemi and Jérôme Euzenat, editors, *Knowledge Engineering: Practice and Patterns, 16th International Conference, EKAW 2008, Acitrezza, Italy, September 29 - October 2, 2008. Proceedings*, volume 5268 of *Lecture Notes in Computer Science*, pages 213–228. Springer, 2008. DOI https://doi.org/10.1007/978-3-540-87696-0_20.

[35] Ondrej Sváb-Zamazal, François Scharffe, and Vojtech Svátek. Preliminary results of logical ontology pattern detection using SPARQL and lexical heuristics. In Eva Blomqvist, Kurt Sandkuhl, François Scharffe, and Vojtech Svátek, editors, *Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009), Washington D.C., USA, 25 October, 2009.*, volume 516 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009. URL http://ceur-ws.org/Vol-516/pap06.pdf.

[36] Christoph Tempich and Raphael Volz. Towards a benchmark for semantic web reasoners - an analysis of the DAML ontology library. In York Sure and Óscar Corcho, editors, *EON2003, Evaluation of Ontology-based Tools, Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools held at the 2nd International Semantic Web Conference ISWC 2003, 20th October 2003 (Workshop day), Sundial Resort, Sanibel Island, Florida, USA*, volume 87 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003. URL http://ceur-ws.org/Vol-87/EON2003_Tempich.pdf.

[37] Christer Thörn, Orjan Eriksson, Eva Blomqvist, and Kurt Sandkuhl. Potentials and limits of graph-algorithms for discovering ontology patterns. In *2005 International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA 2005), International Conference on Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC 2005), 28-30 November 2005, Vienna, Austria*, pages 174–179. IEEE Computer Society, 2005. DOI https://doi.org/10.1109/CIMCA.2005.1631261.

[38] Tania Tudorache, Sean M. Falconer, Csongor Nyulas, Natalya Fridman Noy, and Mark A. Musen. Will semantic web technologies work for the development of icd-11? In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part II*, volume 6497 of *Lecture Notes in Computer Science*, pages 257–272. Springer, 2010. DOI https://doi.org/10.1007/978-3-642-17749-1_17.

[39] Taowei David Wang, Bijan Parsia, and James A. Hendler. A survey of the web ontology landscape. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*, pages 682–694. Springer, 2006. DOI https://doi.org/10.1007/11926078_49.

[40] Patricia L. Whetzel, Natalya Fridman Noy, Nigam H. Shah, Paul R. Alexander, Csongor Nyulas, Tania Tudorache, and Mark A. Musen. Bioportal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. *Nucleic Acids Research*, 39(Web-Server-Issue):541–545, 2011. DOI https://doi.org/10.1093/nar/gkr469.

[41] Zuoshuang Xiang, Jie Zheng, Yu Lin, and Yongqun He. Ontorat: automatic generation of new ontology terms, annotations, and axioms based on ontology design patterns. *Journal of Biomedical Semantics*, 6:4, 2015. DOI https://doi.org/10.1186/2041-1480-6-4.

[42] Mohammed Javeed Zaki. Efficiently mining frequent embedded unordered trees. *Fundamenta Informaticae*, 66(1-2):33–52, Mar/Apr 2005. URL http://content.iospress.com/articles/fundamenta-informaticae/fi66-1-2-03.

[43] Mohammed Javeed Zaki. Efficiently mining frequent trees in a forest: Algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering*, 17(8):1021–1035, 2005. DOI https://doi.org/10.1109/TKDE.2005.125.

Table 9
The largest patterns from a subset of most visited ontologies in BioPortal.

Ontology	Pattern	Size
AERO	$?lhs$ EquivalentTo: ((’has component’ min $?card$ $?classexpr$) and (’has component’ max 1 $?classexpr$) and (’has component’ max 1 $?classexpr$) and (’has component’ max 1 $?classexpr$))	13
OBI	$?lhs$ EquivalentTo: ((’has_specified_output’ some ((’is about’ some $?classexpr$) and $?classexpr$)) and (’has_specified_input’ some $?classexpr$) and (’has part’ some $?classexpr$))	11
ORDO	$?lhs$ SubClassOf: ((’has_birth_prevalence_average_value’ value $?literal$) and (’present_in’ some $?classexpr$) and (’has_prevalence_at_birth_range’ some ’1-9 / 100 000’))	9
ORDO	$?lhs$ SubClassOf: ((’has_point_prevalence_average_value’ value $?literal$) and (’present_in’ some $?classexpr$) and (’has_point_prevalence_range’ some ’1-9 / 100 000’))	9
NCIT	$?lhs$ EquivalentTo: (($?objprop1$ some $?classexpr1$) and ($?objprop2$ some $?classexpr2$) and ($?objprop3$ some $?classexpr3$))	5
PR	$?lhs$ EquivalentTo: ((’only_in_taxon’ some ’Escherichia coli K-12’) and $?classexpr$)	5
PR	$?lhs$ EquivalentTo: ((’only_in_taxon’ some ’Homo sapiens’) and $?classexpr$)	5
PATO	$?lhs$ EquivalentTo: ((’decreased_in_magnitude_relative_to’ some ’normal’) and $?classexpr$)	5
PATO	$?lhs$ EquivalentTo: ((’increased_in_magnitude_relative_to’ some ’normal’) and $?classexpr$)	5
UBERON	$?lhs$ EquivalentTo: ((’part_of’ some $?classexpr$) and $?classexpr$)	4
ZFA	$?lhs$ SubClassOf: (’end stage’ some ’Adult’)	4
ZFA	$?lhs$ SubClassOf: (’end stage’ some ’Hatching:Pec-fin’)	4
ZFA	$?lhs$ SubClassOf: (’end stage’ some ’Unknown’)	4
ZFA	$?lhs$ SubClassOf: (’start stage’ some ’Pharyngula:Prim-5’)	4
ZFA	$?lhs$ SubClassOf: (’start stage’ some ’Segmentation:10-13 somites’)	4
ZFA	$?lhs$ SubClassOf: (’start stage’ some ’Unknown’)	4
GO	$?lhs$ SubClassOf: (’negatively regulates’ some $?classexpr$)	3
GO	$?lhs$ SubClassOf: (’positively regulates’some $?classexpr$)	3
GO	$?lhs$ SubClassOf: (’part of’ some $?classexpr$)	3
GO	$?lhs$ SubClassOf: (’regulates’ some $?classexpr$)	3
EDAM	$?lhs$ SubClassOf: (’is identifier of’ some $?classexpr$)	3
EDAM	$?lhs$ SubClassOf: (’has output’ some $?classexpr$)	3
EDAM	$?lhs$ SubClassOf: (’has topic’ some $?classexpr$)	3
EDAM	$?lhs$ SubClassOf: (’is format of’ some $?classexpr$)	3
EDAM	$?lhs$ SubClassOf: (’has input’ some $?classexpr$)	3
NIFSUBCELL	$?lhs$ SubClassOf: (’proper_part_of’ some $?classexpr$)	3

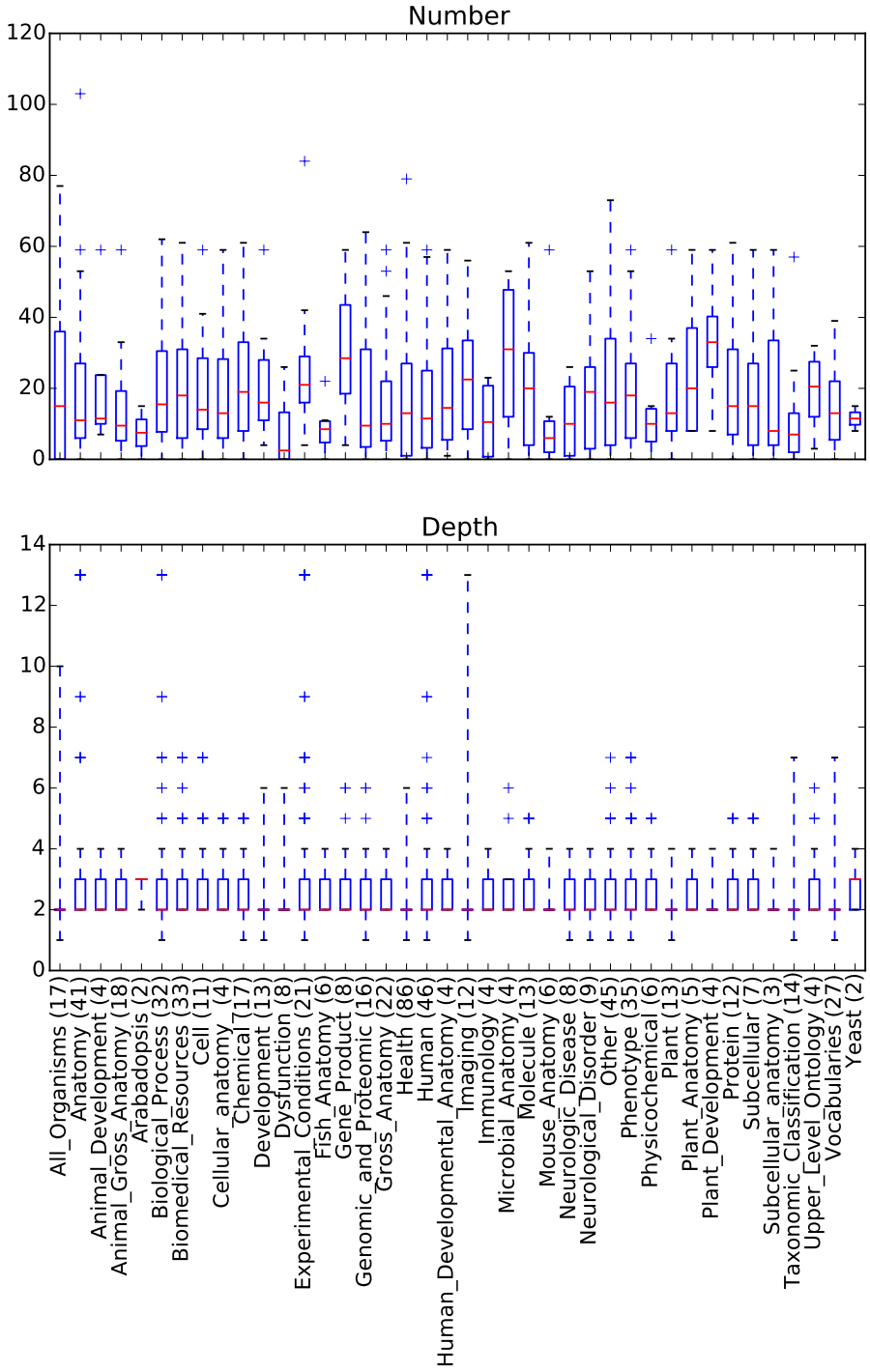


Fig. 18. Statistics for patterns discovered in various categories of ontologies with respect to the number of ontologies in a given topical category and ontology depth.

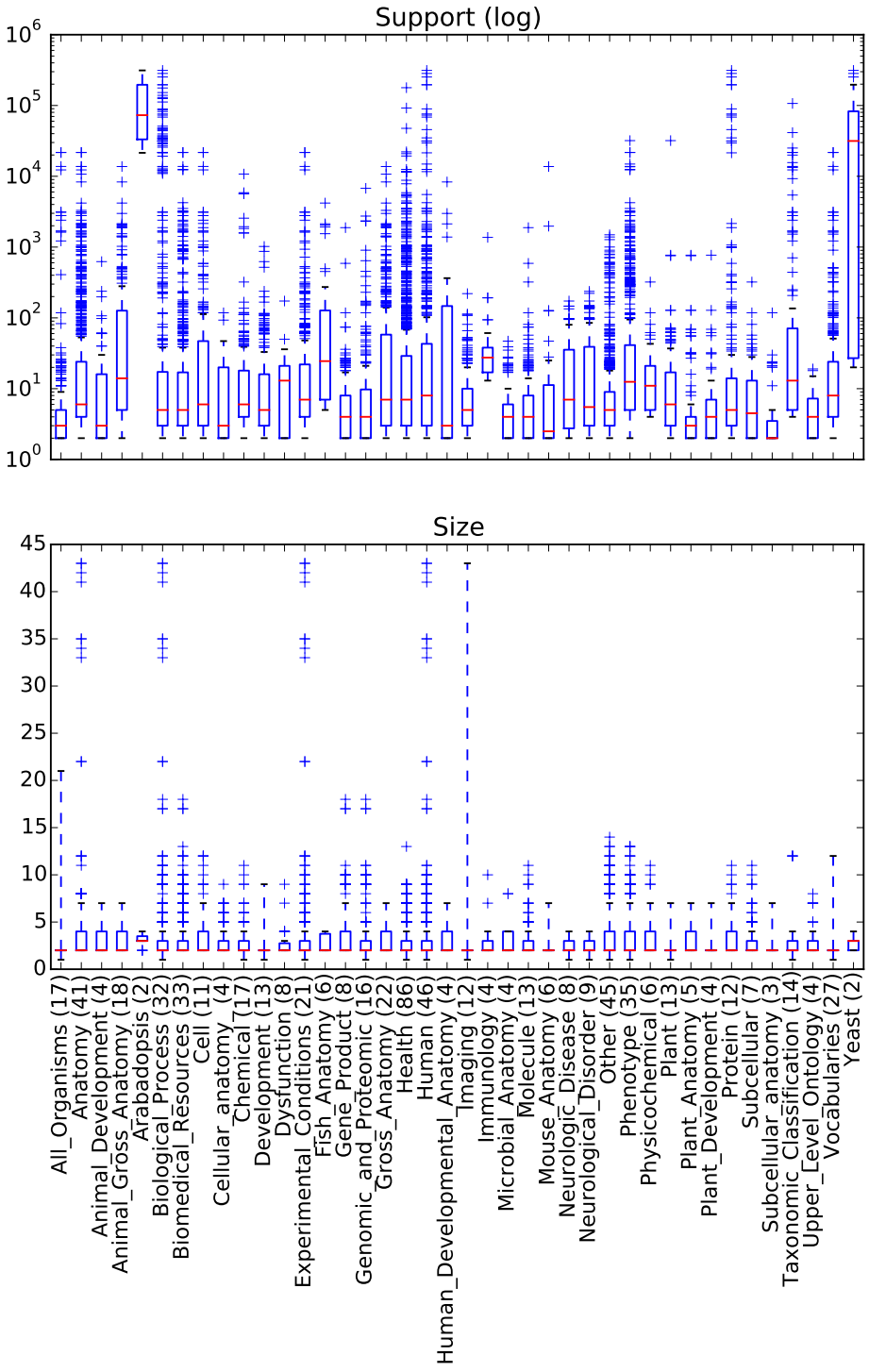


Fig. 19. Statistics for patterns discovered in various categories of ontologies with respect to the support and size of ontologies in a given category.

Table 10

Selected top patterns discovered on the set of BioPortal ontologies, sorted by descending size and support $\sigma_{\mathcal{F}}$

Size	$\sigma_{\mathcal{F}}$	Fragment (URIs and labels)	Ontologies
12	14	obo:IAO_0000078 EquivalentTo {obo:IAO_0000124, obo:IAO_0000423, obo:IAO_0000125, obo:IAO_0000122, obo:IAO_0000123, obo:IAO_0000428, obo:IAO_0000120, obo:IAO_0000121, obo:IAO_0000002} 'curation status specification' EquivalentTo {'uncurated', 'to be replaced with external ontology term', 'pending final vetting', 'ready for release', 'metadata incomplete', 'requires discussion', 'metadata complete', 'organizational term', 'example to be eventually removed'}	BCO, ERO, IAO, OBCS, OBI_BCGO, OBIB, OBI, OBIWS, ODNAE, OGSF, OPL, PCO, SDO, STATO
9	4	obo:OBI_0600047 SubClassOf obo:OBI_0000293 some ((obo:CHEBI_16991 or obo:CHEBI_33697 or obo:PR_000000001) and $?classexpr$) 'sequencing assay' SubClassOf 'has_specified_input' some (('deoxyribonucleic acid' or 'ribonucleic acid' or 'protein') and $?classexpr$)	BCO, OBI_BCGO, OBI, STATO
8	9	obo:IAO_0000225 EquivalentTo {obo:IAO_0000227, obo:IAO_0000228, obo:IAO_0000226, obo:IAO_0000103, obo:IAO_0000229} 'obsolescence reason specification' EquivalentTo {'terms merged', 'term imported', 'placeholder removed', 'failed exploratory term', 'term split'}	BCO, ERO, IAO, OBCS, OBIB, OBI, OPL, PCO, SDO
8	9	span:ProcessualEntity EquivalentTo {span:Process or span:FiatProcessPart or span:ProcessAggregate or span:ProcessBoundary or span:ProcessualContext or span:ProcessualEntity} 'processual_entity' EquivalentTo 'process' or 'fiat_process_part' or 'process_aggregate' or 'process_boundary' or 'processual_context' or 'processual_entity'	ADAR, ADO, BFO, CAO, ERO, HUPSON, OPL, PCO, SDO
8	5	obo:IAO_0000007 SubClassOf BFO_0000051 only (not (obo:IAO_0000005 or obo:IAO_0000104)) 'action specification' SubClassOf 'has part' only (not ('objective specification' or 'plan specification'))	BCO, OBCS, OBI_BCGO, OBIB, STATO
8	4	obo:OBI_0666667 SubClassOf obo:OBI_0000293 some (obo:OBI_0100026 or obo:OBI_0100060 or obo:OBI_0000671) 'nucleic acid extraction' SubClassOf 'has_specified_input' some ('organism' or 'cultured cell population' or 'sample from organism')	OBI_BCGO, OBIB, OBI, STATO
7	9	snap:SpatialRegion EquivalentTo (snap:OneDimensionalRegion or snap:ThreeDimensionalRegion or snap:TwoDimensionalRegion or snap:ZeroDimensionalRegion) 'spatial_region' EquivalentTo ('one_dimensional_region' or 'three_dimensional_region' or 'two_dimensional_region' or 'zero_dimensional_region')	ADAR, ADO, BFO, CAO, ERO, HUPSON, OPL, PCO, SDO
7	6	obo:OBI_0000659 EquivalentTo ((obo:OBI_0000417 some obo:OBI_0000684) and obo:OBI_0000011) 'specimen collection process' EquivalentTo (('achieves_planned_objective' some 'specimen collection objective') and 'planned process')	BCO, OBCS, OBI_BCGO, OBIB, OBI, STATO
7	6	obo:OBI_0100026 EquivalentTo (obo:NCBITaxon_10239 or obo:NCBITaxon_2759 or obo:NCBITaxon_2 or obo:NCBITaxon_2157) 'organism' EquivalentTo ('Viruses' or 'Eukaryota' or 'Bacteria' or 'Archaea')	OBCS, OBI_BCGO, OBIB, OBI, OBIWS, STATO
7	5	obo:OBI_0000453 SubClassOf obo:BFO_0000054 only (obo:OBI_0000299 some obo:IAO_0000109) 'measure function' SubClassOf 'realized in' only (' has_specified_output' some 'measurement datum')	OBCS, OBI_BCGO, OBIB, OBI, STATO
7	5	obo:OBI_0000973 SubClassOf (obo:IAO_0000136 some obo:SO_0000001 and obo:IAO_0000109) 'sequence data' SubClassOf (('is about' some 'region') and 'measurement datum')	OBCS, OBI_BCGO, OBI, OBIWS, STATO
7	5	obo:OBI_0000047 EquivalentTo ((obo:OBI_0000312 some obo:OBI_0000094) and obo:BFO_0000040) 'processed material' EquivalentTo (('is_specified_output_of' some 'material processing') and 'material entity')	OBCS, OBI_BCGO, OBIB, OBI, STATO
7	4	obo:CL_0000151 EquivalentTo ((obo:RO_0002215 some obo:GO_0032940 and obo:CL_0000003) 'secretory cell' EquivalentTo (('capable_of' some 'material processing') and 'native cell')	CL, OBI_BCGO, TAO, VSAO
7	4	obo:IAO_0000015 EquivalentTo (obo:BFO_0000059 some obo:IAO_0000030 and obo:BFO_0000019) 'information carrier' EquivalentTo (('concretizes' some 'information content entity') and 'quality')	OBCS, OBI_BCGO, OBIB, STATO