Syntactic Regularities in OWL Ontologies based on Language Abstractions

Christian Kindermann, Bijan Parsia, and Uli Sattler

University of Manchester, Oxford Rd, Manchester M13 9PL, UK
{christian.kindermann,bijan.parsia,uli.sattler}@manchester.ac.uk

Abstract. OWL ontologies are built and maintained on the basis of all sorts of methods and methodologies using a wide range of tools. As ontologies are primarily published as sets of axioms, its underlying design principles remain opaque. However, a principled and systematic ontology design is likely to be reflected in regularities for axioms. Identifying such regularities may help to recover and unveil conscious design choices and otherwise recurring modelling practices. In this work, we propose a framework for identifying syntactic regularities for axioms in an automated manner. Using this framwork to survey ontologies indexed in BioPortal, we find that most axioms conform to a few number of syntactic regularities. Since conceptual entities are often represented by more than one axiom in OWL, we also motivate a notion for syntactic regularities over sets of axioms. For sets of axioms, an ontology's design appears more variegated. Still, we can often identify a small number of prevalent regularities.

1 Introduction

OWL ontologies are built and maintained on the basis of all sorts of methods and methodologies using a wide range of tools. A basic level of interoperability and reuse is guaranteed by the W3C standards for the Semantic Web.[1] Here, an OWL ontology is defined to consist of two main components: a set of axioms, on the one hand, and a set of imported ontologies, on the other hand. In theory, ontology imports can be used as a basic means to structure an ontology in a meaningful way. Yet, in practice, ontology imports are not used to divide an ontology into parts of fine granularity. As a result, underlying design principles and consistently used modelling techniques remain opaque in published ontologies.

However, a principled and systematic ontology design is likely to be reflected in regularities for axioms. Identifying such regularities may help recover and unveil conscious design choices and otherwise recurring modelling practices. This arguably helps with ontology comprehension and quality assurance in practice.

In this work, we propose a framework for identifying regularities for axioms in an automated manner.The contributions are as follows: (i) we develop a formal framework for discovering syntactic regularities in OWL ontologies, (ii) we propose metrics to qualify discovered regularities and to characterise an ontology's overall design, (iii) we survey ontologies indexed in BioPortal using our framework and metrics.

[1] `https://www.w3.org/standards/semanticweb/`

2 Preliminaries

We adopt the terminology and notation introduced in [1] but adapt definitions for our purposes as needed.

Graphs, Graph Minors, and Graph Isomorphisms A *graph* G is an ordered pair (V, E) where V is a set of *vertices*, and $E \subseteq \{\{v_1, v_2\} \mid (v_1, v_2) \in V \times V \wedge v_1 \neq v_2\}$ is a set of *edges*. A *contraction* of an edge $e = \{v_1, v_2\} \in E$ denotes the removal of e from E and the replacement of both v_1 and v_2 by a single node v' s.t. any node adjacent to either v_1 or v_2 is adjacent to v'.

A *minor* of a graph is a graph obtained by repeatedly contracting edges, removing edges, or removing vertices without adjacent nodes. A *proper minor* is a minor other than the graph itself. A *graph isomorphism* between two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is a bijection $f\colon V_1 \to V_2$ s.t. $\{v_1, v_1'\} \in E_1$ iff $\{f(v_1), f(v_1')\} \in E_2$.

Trees, Terms, and Term Trees A *tree* is a connected acyclic undirected graph. We denote by $\mathbb{N}^*$ the set of finite strings over the natural numbers $\mathbb{N}$ (we assume $0 \in \mathbb{N}$). A *ranked alphabet* is a pair $(\mathcal{F}, \mathsf{Arity})$ where $\mathcal{F}$ is a finite set of symbols and Arity is a function from $\mathcal{F}$ into $\mathbb{N}$. The *arity* of a symbol $f \in \mathcal{F}$ is $\mathsf{Arity}(f)$. The set of symbols of arity p is denoted by $\mathcal{F}_p$. Elements of arity $0, 1, \ldots, p$ are respectively called constants, unary, ..., p-ary symbols. Let $\mathcal{X}$ be a set of constants called *variables*. We assume F_0 and $\mathcal{X}$ to be disjoint. An *unranked alphabet* $\mathcal{U}$ is a set of symbols that do not have a fixed arity. The set of *terms* $T(\Sigma, \mathcal{X})$ over the alphabet $\Sigma = \mathcal{U} \cup \mathcal{F}$ and the set of variables $\mathcal{X}$ is the smallest set defined inductively as follows:

- $F_0 \subseteq T(\Sigma, \mathcal{X})$,
- $\mathcal{U} \subseteq T(\Sigma, \mathcal{X})$,
- $\mathcal{X} \subseteq T(\Sigma, \mathcal{X})$, and
- if $p \geq 1, f \in F_p \cup \mathcal{U}$ and $t_1, \ldots, t_p \in T(\Sigma, \mathcal{X})$, then $f(t_1, \ldots, t_p) \in T(\Sigma, \mathcal{X})$.

If $\mathcal{X} = \emptyset$ then $T(\Sigma, \mathcal{X})$ is also written $T(\Sigma)$. Terms in $T(\Sigma)$ are called *ground terms*.

A *term tree* t is a finite ordered tree over Σ that is defined as a partial function $t\colon \mathbb{N}^* \to \Sigma$ with domain written $Pos(t)$ satisfying the following properties

- $Pos(t)$ is finite, non-empty, and prefix-closed,
- for all $p \in Pos(t)$
  - if $t(p) \in \mathcal{X} \cup F_0$, then $\{j \mid pj \in Pos(t)\} = \emptyset$
  - if $t(p) \in \mathcal{F}_n$, then $\{j \mid pj \in Pos(t)\} = \{1, \ldots, n\}$
  - if $t(p) \in \mathcal{U}$, then
    - $\{j \mid pj \in Pos(t)\} = \{1, \ldots, k\}$ for some $k \geq 1$, or
    - $\{j \mid pj \in Pos(t)\} = \emptyset\}$.

A term in $T(\Sigma, \mathcal{X})$ may be viewed as a term tree, the leaves of which are labeled with variables or constant symbols and internal nodes are labeled with symbols of positive arity with out-degree equal to the arity of the label. In the following we do not distinguish between terms and their corresponding term trees.

A *subterm* $t_{|p}$ of a term $t \in T(\Sigma, \mathcal{X})$ at position p is defined as following:

– $Pos(t_{|p}) = \{q \mid pq \in Pos(t)\}$,
– for $q \in Pos(t_{|p})$ we define $t_{|p}(q) = t(pq)$.

We use $t[u]_p$ for the term obtained by replacing the subterm $t_{|p}$ by u in term t.

Substitutions A *substitution* σ is a function from $\mathcal{X}$ into $T(\Sigma, \mathcal{X})$ where there are only finitely many variables not mapped to themselves. The *domain* of a substitution σ is the subset of variables $x \in \mathcal{X}$ such that $\sigma(x) \neq x$. The substitution $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is the identity on $\mathcal{X} \setminus \{x_1, \ldots, x_n\}$ and maps $x_i \in \mathcal{X}$ on $t_i \in T(\Sigma, \mathcal{X})$ for every index $1 \leq i \leq n$. Substitutions are extended to $f(t_1, \ldots, t_n) \in T(\Sigma, \mathcal{X})$ by $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_k))$.

In the following, we do not distinguish between a substitution and its extension to $T(\Sigma, \mathcal{X})$, and as usual, we will often use substitutions in postfix notation, i.e., $t\sigma$ is the result of applying σ to the term t.

Contexts A term $C \in T(\Sigma, \mathcal{X})$ with variables is called a *context* and we write $C[x_1, \ldots, x_n]$ to denote a context C with variables $x_1, \ldots, x_n$. We write $C[t_1, \ldots, t_n]$ to denote the term $C\sigma \in T(\Sigma)$ where σ is the substitution $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$.

Tree Transducers We define transformations on trees as finite bottom-up tree transducers. A *tree transducer* is a tuple $A = (Q, \Sigma, \Sigma', Q_f, \Delta)$ where Σ is the input alphabet, Σ' is the output alphabet, Q is a set of (unary) states, $Q_f \subseteq Q$ is a set of final states, and Δ is a set of transduction rules for trees.

For the purpose of treating ranked and unranked symbols in a uniform manner, we describe the left-hand side of transduction rules by $f(h)$, where f is a symbol in Σ and h is a (finite) sequence of symbols. To ensure that h is of suitable length, we condition each transduction rule to a language $\mathcal{H}$, called a *horizontal language*[2] that formulates constraints for h as needed. With this, a transduction rule in Δ is of the following form:

$$f(h) \to q(\phi(h)) \text{ subject to } h \in \mathcal{H},$$

where $f \in \Sigma, q \in Q$, $\mathcal{H}$ a horizontal language, and ϕ is a function $\mathcal{H} \to T(\Sigma', \mathcal{X})$. Note, that each transduction rule has a state q as the root of the tree on its right-hand side. Therefore we can assume h to be a sequence of states or the empty tree in case of rules for leafs. To make this precise, we restrict horizontal languages $\mathcal{H}$ as follows:

$$\begin{aligned}\mathcal{H} \subseteq \mathcal{H}_Q = \{\varepsilon\} \cup \{q_1(x_1) \cdots q_n(x_n) \mid\ & n \in \mathbb{N},\\ & q_1, \ldots, q_n \in Q,\\ & x_1, \ldots, x_n \in \mathcal{X},\\ & x_i \neq x_j \text{ for } i, j \in \{1, \ldots, n\} \text{ and } i \neq j\}.\end{aligned}$$

So $h \in \mathcal{H}_Q$ is a (finite) sequence of (single variable) contexts $q_i(x_i)$. Each context in such a sequence can is uniquely identified by its variable x_i. Note, however, that the

[2] The notion of a horizontal language is closely related to the notion of *hedge language*s for unranked symbols in the literature. However, as we work with both ranked and unranked symbols, we decided against overloading this commonly used terminology to avoid confusion.

states of contexts $q_i(x_i)$ and $q_j(x_j)$ with $x_i \neq x_j$ in a given sequence can be identical. Transduction rules with $\mathcal{H} = \{\varepsilon\}$ can be applied to the leaves of a tree will be written $a \to q(\phi(a))$ for $a \in \Sigma$.

In the following, we assume horizontal languages to be regular languages. Furthermore, we will use regular expressions to define them. We assume regular expressions using the Kleene star to introduce fresh individuals as required. For example $q(x_1)q(x_2) \in Q(x)^*$ but $q(x_1)q(x_1) \notin Q(x)^*$ (assuming $q \in Q$). Furthermore, we sometimes use commas to separate distinct elements of a sequences h. For example, $f(h) = f(q_1(x_1)q_2(x_2)q_3(x_3))$ may also be written $f(q_1(x_1), q_2(x_2), q_3(x_3))$ and vice-versa with no difference in meaning whatsoever.

Let $t, t' \in T(\Sigma \cup \Sigma' \cup Q)$. The move relation $\to_A$ for a tree transducer A is defined by

$$t \to_A t' \Leftrightarrow \begin{cases} \exists f(h) \to q(\phi(h)) \text{ subject to } h \in \mathcal{H} \in \Delta\colon \\ \exists q_1(x_1) \cdots q_n(x_n) = h \in \mathcal{H}\colon \\ \exists u_1, \ldots, u_n \in T(\Sigma')\colon \\ \exists C \in \mathcal{C}(\Sigma \cup \Sigma' \cup Q)\colon \\ t = C[f(q_1(u_1), \ldots, q_n(u_n))] \wedge \\ t' = C[q(\phi(h)\{x_1 \to u_1, \ldots, x_k \to u_k\})] \end{cases}$$

The reflexive and transitive closure of $\to_A$ is $\to_A^*$. A transduction from a ground term $t \in T(\Sigma)$ to a ground term $t' \in T(\Sigma')$ is a sequence of move steps of the form $t \to_A^* q(t')$, such that q is a final state. The relation between terms $T(\Sigma)$ and $T(\Sigma')$ induced by A is the relation defined by

$$A^* = \{(t, t') \mid t \to_A^* q(t'), t \in T(\Sigma), t' \in T(\Sigma'), q \in Q_f\}.$$

A tree transducer is *deterministic* if there are no rules with the same left-hand side. In the following, we assume transducers to be deterministic and we write $A(t) = t'$ if $(t, t') \in A^*$.

OWL The Web Ontology Language (OWL) is defined defined in terms of a structural specification.[3] This specification, however, does not define a formal language in the conventional sense. Rather, it specifies structural features that a formal language needs to represent (in whatever way) to be considered an OWL language. In this work, we pick a concrete OWL syntax, namely the functional-style syntax, as a representative for OWL.[4] So, an *OWL axiom* is an axiom as specified by OWL's structural specification[5] written in functional-style syntax. An *OWL ontology* is a finite set of OWL axioms. Please note that this notion of an OWL ontology does, strictly speaking, not correspond the definition of an OWL ontology in OWL's structural specification. Also, we disregard axiom annotations in all aspects.

[3] `https://www.w3.org/TR/owl2-syntax/`

[4] Note that the particular choice of a representative language for OWL is immaterial for the remainder of this work. In the following, we discuss OWL axioms exclusively on the basis of OWL's structural specification.

[5] `https://www.w3.org/TR/owl2-syntax/#Axioms`

3 Language Abstraction

We understand the notion of *abstraction* intuitively as a vehicle for analysing a subject of interest w.r.t. some aspects while disregarding others. As such, an abstraction involves a conceptualisation of the subject that is based on the removal of some level of detail. This removal of information is warranted if the removed details are immaterial for the subsequent analysis. Thus, we can characterise and describe an abstraction both in terms of what kind of information is preserved and what kind of information is discarded. We use this intuitive understanding of the notion of abstraction to motivate a (formal) notion for abstracting from a formal language into another.

3.1 Abstraction for Formal Languages

Following the intuition that the notion of abstraction is based on the removal of some information, we propose a notion of abstraction for formal languages that captures this intuition by syntax-directed transformations.

Definition 1 (Language Abstraction). *An* abstraction *for a language $\mathcal{L}$ over finite alphabet Σ into a language $\mathcal{L}_a$ over finite alphabet Σ_a is defined by a tree transducer $A = (Q, \Sigma, \Sigma_a, Q_f, \Delta)$ such that:*

(i) for all $t \in \mathcal{L}$ there exists a $A(t) \in \mathcal{L}_a$,
(ii) there exist $t, t' \in \mathcal{L}$ s.t. $t \neq t'$ with $A(t) = A(t')$,
(iii) for $t \in \mathcal{L}$ there exists a minor t_m that is isomorphic to $A(t)$.

A language abstraction can be seen as a function from one formal language into another. Note that this function involves a loss of information since there are at least two distinct elements in the original language that are indistinguishable under the abstraction (see condition (ii) in Definition 1).

Also note that a language abstraction allows for the removal of two distinct kinds of information. On the one hand, an abstraction can remove structural information by transforming a tree following operations of graph minors. And on the other hand, lexical information can be removed by mapping to different symbols in the original language to the same symbol.

Example 1 (Language Abstraction for OWL). Consider the axioms

$$\texttt{SubClassOf}(\mathsf{A}_1,\texttt{ObjectMinCardinality}(2,\mathsf{S},\mathsf{A}_2))$$
$$\texttt{SubClassOf}(\mathsf{B}_1,\texttt{ObjectSomeValuesFrom}(\mathsf{R},\mathsf{B}_2)).$$

Figure 1 shows their respective term trees.

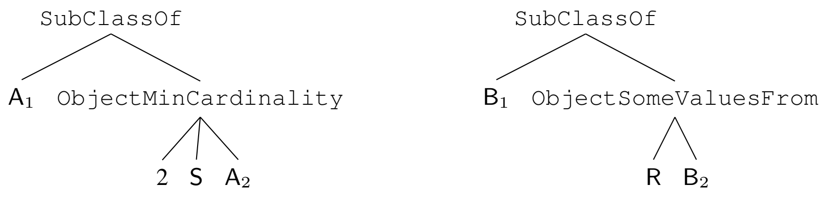

Fig. 1: Structurally different term trees.

Clearly, these axioms differ w.r.t. their syntactical structure. However, they are similar in the sense that they both involve an object property restriction on the right-hand side of an `SubClassOf` axiom. By abstracting over the particular kind of object property restrictions, this similarity can be made explicit. Consider the following language abstraction $A = (Q, \Sigma, \Sigma', Q_f, \delta)$ where

- $Q = \{q, q_{iri}, q_f\}$
- $\Sigma = \Omega \cup IRI$ with Ω being OWL's logical symbols and $IRI = \{\mathsf{A}_1, \mathsf{A}_2, \mathsf{B}_1, \mathsf{B}_2, \mathsf{S}, \mathsf{R}\}$,
- $\Sigma' = (\Sigma \cup \{\texttt{Restriction}\}) \setminus \{\texttt{ObjectSomeValuesFrom}, \texttt{ObjectSomeValuesFrom}\}$
- $Q_f = \{q_f\}$
- $\delta = \{$
  (R1) $x \rightarrow q_{iri}(x)$ subject to $x \in IRI$,
  (R2) $x \rightarrow q_{\mathbb{N}}(x)$ subject to $x \in \mathbb{N}$,
  (R3) $\texttt{SubClassOf}(q(x), q(y)) \rightarrow q_f(\texttt{SubClassOf}(x, y))$ subject to $q(x)q(y) \in q(T(\Sigma')q(\Sigma'))$,
  (R4) $\texttt{ObjectSomeValuesFrom}(q_{iri}(x), q_{iri}(y)) \rightarrow q(\texttt{Restriction}(x, y))$ subject to $q_{iri}(x)q_{iri}(y) \in q_{iri}(IRI)q_{iri}(IRI)$,
  (R5) $\texttt{ObjectMinCardinality}(q_{\mathbb{N}}(x), q_{iri}(y), q_{iri}(z)) \rightarrow q(\texttt{Restriction}(y, z))$ subject to $q_{\mathbb{N}}(x)q_{iri}(x)q_{iri}(y) \in q_{\mathbb{N}}(\mathbb{N})q_{iri}(IRI)q_{iri}(IRI)$
  $\}$.

Using language abstraction A for the two axioms shown in Figure 1 results in the following structurally similar trees (in the more abstract target language):

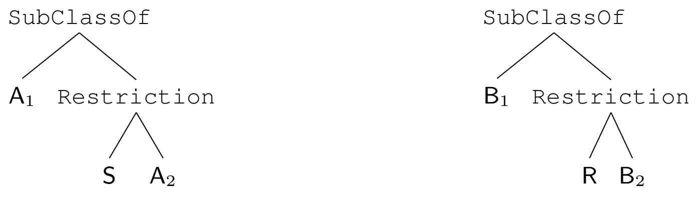

Fig. 2: Structural similarities made explicit by language abstraction A.

3.2 Detailed Demonstration of Language Abstractions

While the notion of language abstractions is based on the intuitively understandable idea of rewriting syntax trees by removing and renaming nodes and edges, we acknowledge that the corresponding formalism using tree transducers is technically involved. In this section, we present a fully worked-out example to demonstrate and elucidate the intricacies of our approach.

In Example 1 of Section 3, we already presented a language abstraction for OWL that transforms `ObjectSomeValuesFrom` and `ObjectMinCardinality` restrictions into an (abstract) symbol `Restriction` which is not in OWL. We build on this example and present all steps involved to transform a given OWL axiom into the resulting abstract language using the symbol `Restriction`. To demonstrate how nested logical operators are handled, we extend Example 1 by including transduction rules for `ObjectIntersectionOf`.

Formal Specification of a Language Abstraction We define the language abstraction $A = (Q, \Sigma, \Sigma', Q_f, \Delta)$, where

$$Q = \{\ q_{CE}, q_{PE}, q_{\mathbb{N}}, q_f\},$$

$$\begin{aligned} \Sigma = \{\ & \texttt{SubClassOf}, \\ & \texttt{ObjectIntersectionOf}, \\ & \texttt{ObjectSomeValuesFrom}, \\ & \texttt{ObjectMinCardinality}\} \cup \\ & \mathbb{N} \cup N_C \cup N_R, \end{aligned}$$

$$\begin{aligned} \Sigma' = \ & (\Sigma \setminus \{\texttt{ObjectSomeValuesFrom}\}) \cup \\ & \{\texttt{Restriction}\}, \end{aligned}$$

$$Q_f = \{\ q_f\},$$

$$\begin{aligned} \Delta = \{\ & \mathsf{C} \rightarrow q_{CE}(\phi_{\mathbf{id}}(\mathsf{C})) \text{ subject to } \mathsf{C} \in N_C, \\ & \mathsf{R} \rightarrow q_{PE}(\phi_{\mathbf{id}}(\mathsf{R})) \text{ subject to } \mathsf{R} \in R_C, \\ & n \rightarrow q_{\mathbb{N}}(\phi_{\mathbf{id}}(n)) \text{ subject to } n \in \mathbb{N}, \\ & \texttt{ObjectIntersectionOf}(h) \rightarrow q_{CE}(\phi_{\sqcap}(h)) \text{ subject to } h \in q_{CE}(x_1)q_{CE}(x)^+, \\ & \texttt{ObjectSomeValuesFrom}(h) \rightarrow q_{CE}(\phi_{R1}(h)) \text{ subject to } h \in \{q_{PE}(x)q_{CE}(y)\}, \\ & \texttt{ObjectMinCardinality}(h) \rightarrow q_{CE}(\phi_{R2}(h)) \text{ subject to } h \in q_{\mathbb{N}}(\mathbb{N})q_{PE}(x)q_{CE}(y), \\ & \texttt{SubClassOf}(h) \rightarrow q_f(\phi_{\sqsubseteq}(h)) \text{ subject to } h \in \{q_{CE}(x)q_{CE}(y)\}\}. \end{aligned}$$

In this definition of A, we use N_C and N_R to denote the sets of named classes and property expressions. The functions ϕ in the set of transduction rules Δ are defined as follows:

$$\phi_{\mathbf{id}}\colon T(\Sigma,\mathcal{X})\cup\mathbb{N}\to T(\Sigma',\mathcal{X})\cup\mathbb{N},\quad x\mapsto x,$$

$$\phi_{\sqcap}\colon T(\Sigma'\cup Q,\mathcal{X})T(\Sigma'\cup Q,\mathcal{X})^{+}\to T(\Sigma',\mathcal{X}),\quad q_{CE}(x_1)\dots q_{CE}(x_n)\mapsto \texttt{ObjectIntersectionOf}(x_1\dots x_n),$$

$$\phi_{R1}\colon \mathbb{N}(T(\Sigma'\cup Q,\mathcal{X}))^2\to T(\Sigma',\mathcal{X}),\quad q_{PE}(x)q_{CE}(y)\mapsto \texttt{Restriction}(x,y),$$

$$\phi_{R2}\colon (T(\Sigma'\cup Q,\mathcal{X}))^3\to T(\Sigma',\mathcal{X}),\quad q_{\mathbb{N}(x)},q_{PE}(y),q_{CE}(z)\mapsto \texttt{Restriction}(y,z),$$

$$\phi_{\sqsubseteq}\colon T(\Sigma'\cup Q,\mathcal{X})^2\to T(\Sigma',\mathcal{X}),\quad q_{CE}(x),q_{CE}(y)\mapsto \texttt{SubClassOf}(x,y)$$

In the following, we will refer to the six transduction rules in Δ by (R1)–(R7) in the order they are presented.

We present an exmaple transduction of the OWL axiom shown as a (term) tree in Figure 3 below.

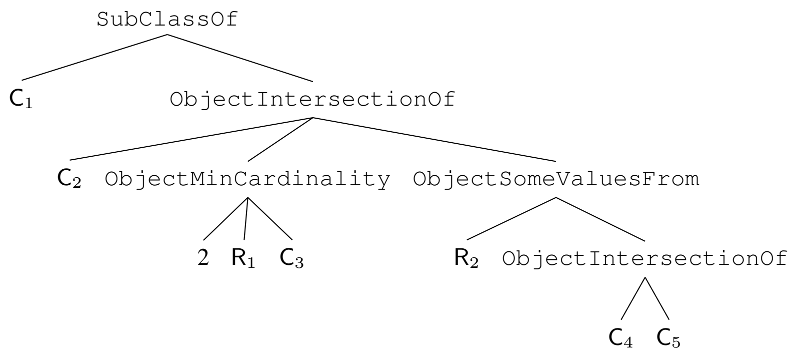

Fig. 3: Example term tree.

Before we discuss how the move relation $\to_A$ works on a technical level, we present how transduction rules effectively change the structure of a given tree on a more intuitive level by way of visualisation.

3.3 Visualisation of Tree Transformation

First, only transduction rules for leafs, i.e., for named classes, object properties, and numbers are applicable. Figure 4 shows the resulting tree after after all rules for leafs are applied to the example tree shown in Figure 3. We colour newly added nodes in blue. In this case, the rules for leaf nodes only introduced states as parent nodes for leafs.

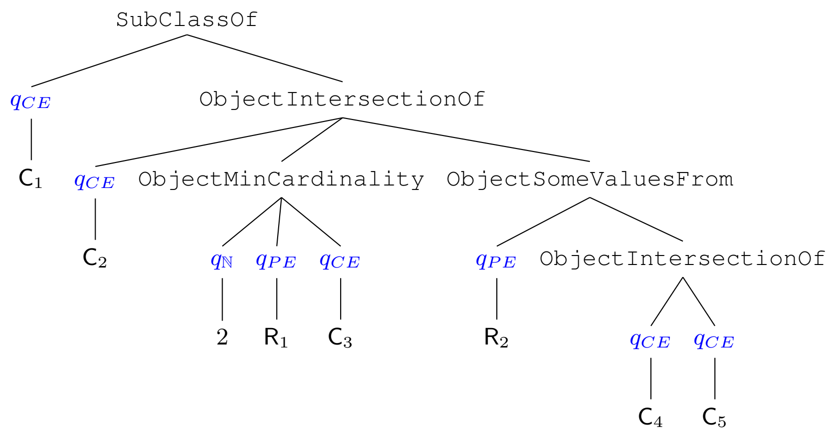

Fig. 4: Example tree $t \overset{*}{\rightarrow}_A t_1$ after (all possible) applications of rules (R1)–(R3) for leaf nodes.

Next, we choose to apply rule (R4) for the symbol `ObjectIntersectionOf` to the right-most subtree.

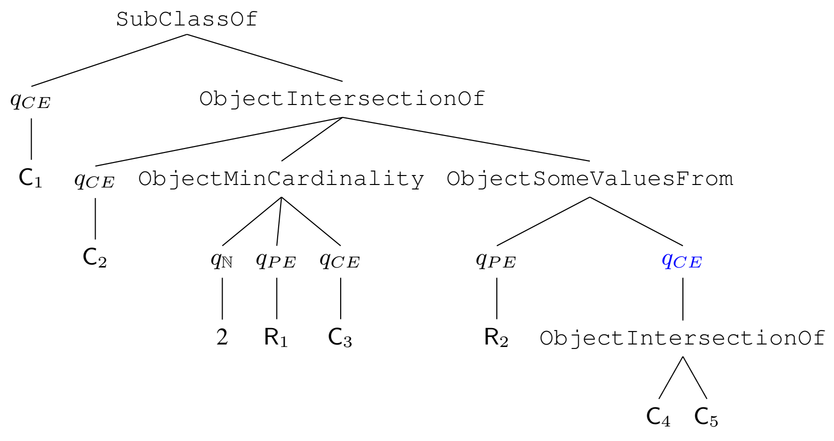

Fig. 5: Example tree $t_1 \rightarrow_A t_2$ after application of rule (R4) for `ObjectIntersectionOf`.

Next, we can apply the rule (R5) for the symbol `ObjectSomeValuesFrom`.

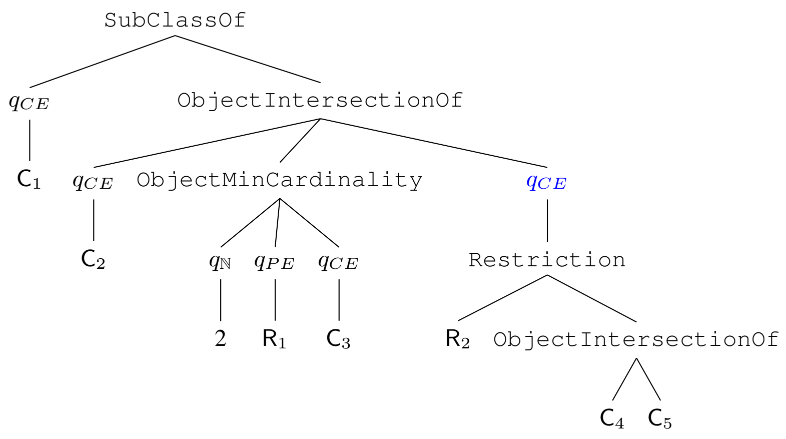

Fig. 6: Example tree $t_2 \rightarrow_A t_3$ after application of rule (R5) for `ObjectSomeValuesFrom`.

Likewise, we proceed with rule (R6) for the symbol `ObjectMinCardinality`.

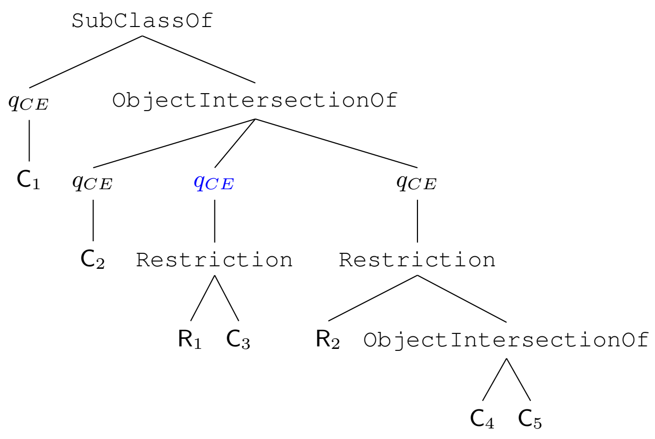

Fig. 7: Example tree $t_3 \rightarrow_A t_4$ after application of rule (R6) for symbol `ObjectMinCardinality`.

Next, rule (R4) for symbol `ObjectIntersectionOf` (at a different position) is applicable again:

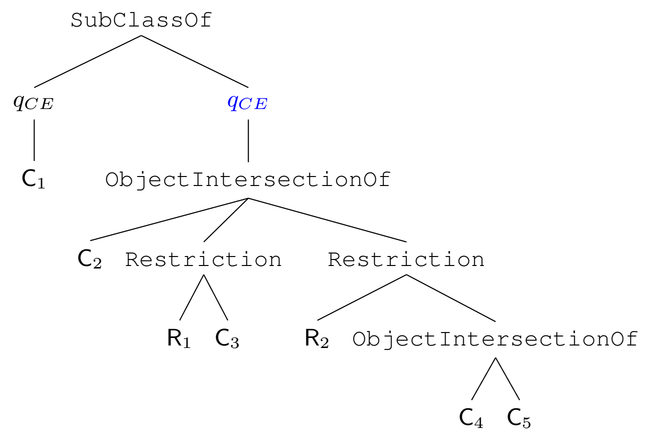

Fig. 8: Example tree $t_4 \rightarrow_A t_5$ after application of rule (R4) for the symbol `ObjectIntersectionOf`.

And lastly, rule (R7) can be applied to the node `SubClassOf`:

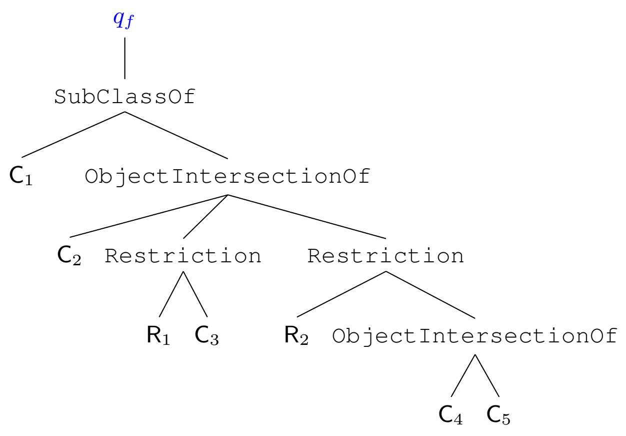

Fig. 9: Example tree $t_4 \rightarrow_A t_5$ after application of (R7) for the symbol `SubClassOf`.

3.4 Move Relation on a Technical Level

The example tree transduction shown by way of visualisations in the Section 3.3 does not show all of the technical details involved in a single application of the move relation $\rightarrow_A$. In this section, we highlight and discuss these details by way of example for the application of the move relation to tree t_3 shown in Figure 6 to yield tree t_4 shown in Figure 7. We repeat tree t_3 in the figure below for convenience.

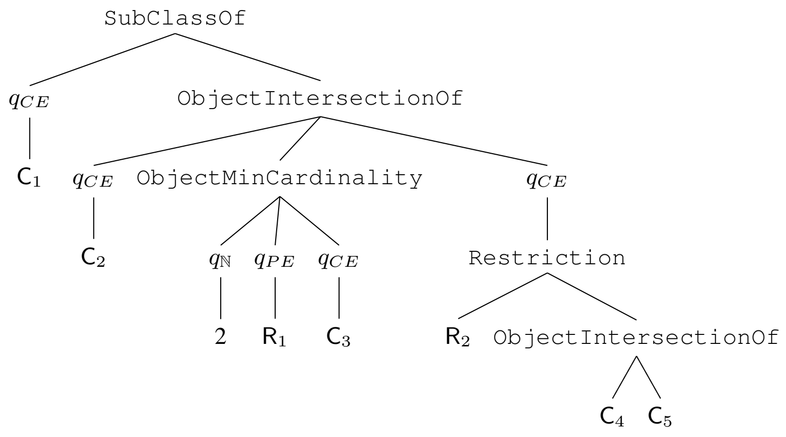

Fig. 10: Example tree t_3.

In order to move from tree t_3 to t_4, we need to apply rule (R6), namely

$$\texttt{ObjectMinCardinality}(h) \to q_{CE}(\phi_{R2}(h)) \text{ subject to } h \in q_{\mathbb{N}}(\mathbb{N})q_{PE}(x)q_{CE}(y)$$

To apply this rule to a tree, the following criteria of the move relation $\to_A$ need to be satisfied:

1. there need to exist $u_1, u_2, u_3 \in T(\Sigma')$
2. such that there exists a context C for which

$$t_3 = C[\texttt{ObjectMinCardinality}(q_{\mathbb{N}}(u_1), q_{PE}(u_2), q_{CE}(u_3))]$$

   holds.

First, we note that $2, R_1, C_3 \in \Sigma'$ which are possible candidates for u_1, u_2 and u_2 respectively. Next, recall that a context $C \in \mathcal{C}(\Sigma \cup \Sigma' \cup Q)$ is a context with a *single* variable. Also recall, that $C[t]$ denotes a term in $T(\Sigma \cup \Sigma' \cup Q)$ obtained from C by replacing its variable by t. Now, if we define C as shown in Figure 11, then $t_3 = C[\texttt{ObjectMinCardinality}(q_{\mathbb{N}}(2), q_{PE}(R_1), q_{CE}(C_3))]$

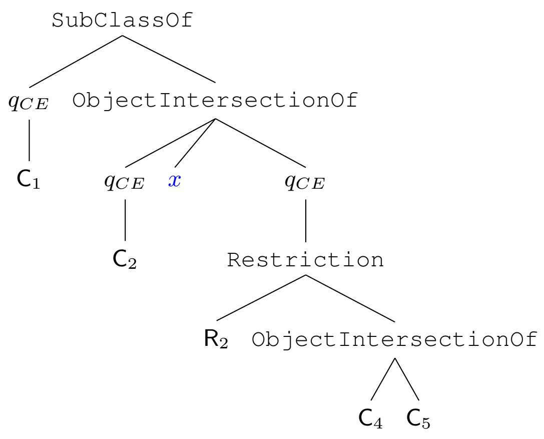

Fig. 11: Context C.

At this point, it is instructive to summarise all relevant elements discussed so far w.r.t. to the definition of $\rightarrow_A$:

Definition of $t \rightarrow_A t'$	Use of $\rightarrow_A$ w.r.t. rule (R6) applied to t_3 to yield t_4
$\exists f(h) \rightarrow q(\phi(h)) \in \Delta$:	$(R6)$
$\exists q_1(x_1) \cdots q_n(x_n) = h$	$q_{\mathbb{N}}(2) q_{PE}(x) q_{CE}(y)$
$\exists u_1, \ldots, u_n \in T(\Sigma')$:	$u_1 = 2, u_2 = R_1, u_3 = C_3$
$\exists C \in \mathcal{C}(\Sigma \cup \Sigma' \cup Q)$:	See Figure 11
$t = C[f(q_1(u_1), \ldots, q_n(u_n))] \wedge$	$t_3 = C[\mathtt{ObjectMinCardinality}(q_{\mathbb{N}}(2), q_{PE}(R_1), q_{CE}(C_3))] \wedge$
$t' = C[q(\phi(h)\{x_1 \rightarrow u_1, \ldots, x_k \rightarrow u_k\})]$	$t_4 = ?$

So far, we have only discussed the left-hand side of $t \rightarrow_A t'$ w.r.t. rule (R6). The right-hand side, is defined in $t \rightarrow_A t'$ as

$$C[q(\phi(h)\{x_1 \rightarrow u_1, \ldots, x_k \rightarrow u_k\})].$$

Note, how all elements used to define t' are already specified. In particular, we have

- C as in Figure 11,
- q as q_{CE} specified by the right-hand side of rule (R6),
- $\phi(h)$ as $\phi_{\geq 1}(h)$ specified by the right-hand side of rule (R6),
- h as $q_{\mathbb{N}}(2) q_{PE}(x) q_{CE}(y)$, and
- $u_1 = 2$, $u_2 = R_1$ as well as $u_3 = C_3$.

With this, we can evaluate the right-hand side of the move relation $\rightarrow_A$ in a straightforward manner as follows:

- $\phi_{R2}(q_{\mathbb{N}}(x) q_{PE}(y) q_{CE}(z)) = \mathtt{Restriction}(y, z)$,
- $\mathtt{Restriction}(y, z)\{y \mapsto u_2, z \mapsto u_3,\} = \mathtt{Restriction}(u_2, u_3)$
  $= \mathtt{Restriction}(R_1, C_3)$

– and finally, $C[q_{CE}(\texttt{Restriction}(R_1, C_3))]$ evaluates via substitution to t_4 as shown by Figure 11 as shown below.

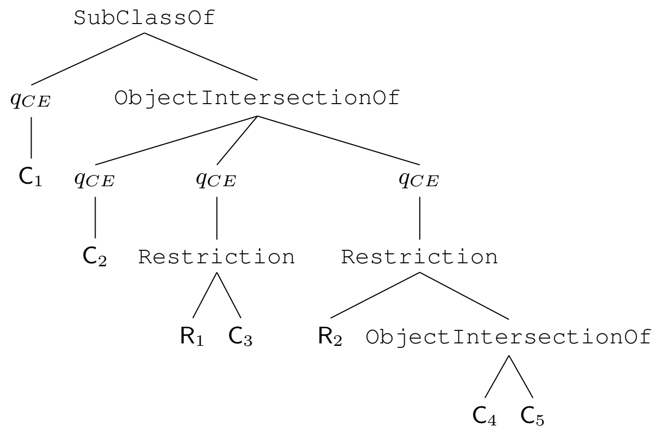

Fig. 12: Example tree t_4 after application of rule (R6).

3.5 Abstraction Algebra

In this section, we briefly touch on the subject of relations between language abstractions. For example, two language abstractions can be related by some form form of generalisation or specialisation or they can be completely orthogonal. To discuss such relations we propose the idea of an algebra for language abstractions. Note, however, that the following presentation is not intended to give a exhaustive or otherwise complete account of relationships for language abstractions.

Definition 2 (Abstraction by Ground Generalisation). *An* abstraction by ground generalisation *w.r.t. a set of constants $C \subseteq \Sigma$ is a language abstraction $A = (Q, \Sigma, \Sigma \cup \mathcal{X}, Q_f, \Delta)$ s.t.*

(i) for $c \in C$ there exists a rule $c \to q(\phi(c)) \in \Delta$, where $\phi(c) = x$ maps a constant to a fresh variable, and
(ii) for $s \notin C$ there exists a rule $f(h) \to q(\phi(h)) \in \Delta$, where $\phi(h) = f(h)$.

An Abstraction by Ground Generalisation removes lexical information by replacing constant symbols with variables.[6] However, all of the syntactical structure pertaining to non-constant symbols is preserved.

A straightforward abstraction that does not preserve the structure of non-constant symbols, simply reduces the number of its children, i.e., immediate subterms.

[6] One can argue that the introduction of different variables for the same constant symbol at different positions in a tree removes structural information. Indeed, one can easily define an abstraction by *renaming* that uniformly replaces a constant at different positions in a tree with the same variable.

Definition 3 (Abstraction by Projection). *A language abstraction* $A = (Q, \Sigma, \Sigma_a, Q_f, \Delta)$ *is a* projection *for symbol* $f \in \Sigma$ *if there exists a rule* $f(h) \to q(\phi(h)) \in \Delta$ *s.t.* $\phi(h) = f'(\phi(h))$ *and* $|\phi(h)| \leq |h|$.

A special case of an abstraction by projection is an abstraction that removes selected symbols together with all of their subterms.

Definition 4 (Abstraction by Selection). *A language abstraction* $A = (Q, \Sigma, \Sigma_a, Q_f, \Delta)$ *is a* selection *for symbols* $S \subseteq \Sigma$ *if for all* $f \notin S$ *there exists a rule* $f(h) \to q(\phi(f(h))) \in \Delta$ *and* $\phi(f(h)) = \varepsilon$.

4 Measuring Syntactic Regularity

We are interested in syntactic *regularities* for axioms in ontologies. We understand syntactic regularities in terms of shared syntactic properties between axioms. Such shared properties can be made precise and explicit by language abstractions as defined in Section 3. The basic idea being that shared syntactic properties are *preserved* under language abstractions while differences are *abstracted away*.

Definition 5 (Syntactic Regularity). *Let* A *be a language abstraction over language* $\mathcal{L}$*. A* syntactic regularity *is an equivalence class* $[t]_A = \{t' \in \mathcal{L} \mid A(t') = A(t)\}$ *for a term* $t \in \mathcal{L}$ *induced by* A.

Elements of a syntactic regularity are (syntactically) indistinguishable under the used abstraction. These elements necessarily share some syntactic aspects due to constraint (iii) in Definition 1 for language abstractions.

We extend the notion of syntactic regularities for elements of a language to *sets* of elements in a straightforward manner.

Definition 6 (Syntactic Regularity Sets). *Let* $\mathcal{S} = \{S_1, \ldots, S_n\}$ *be a family of sets over a language* $\mathcal{L}$*. A* syntactic regularity set $[S_i]_A$ *for* $S_i \in \mathcal{S}$ *w.r.t. a language abstraction* A *is defined by:*

$$\begin{aligned}[S_i]_A = \{S' \in \mathcal{S} \mid\ & \textit{there exist a sequence } A(t_1) \cdots A(t_n) \textit{ with } \{t_1, \ldots, t_n\} = S_i, \\ & \textit{there exist a sequence } A(t'_1) \cdots A(t'_n) \textit{ with } \{t'_1, \ldots, t'_n\} = S', \\ & \textit{s.t.} A(t_1) \cdots A(t_n) = A(t'_1) \cdots A(t'_n)\}.\end{aligned}$$

The amount of distinct syntactic regularities and their respective size shed light on whether the ontology is designed in a uniform manner. We refer to regularities of comparatively large size as *prevalent*.

Definition 7 (Prevalent Syntactic Regularity). *A syntactic regularity* $[t]_A$ *in a language* $\mathcal{L}$ *w.r.t. a language abstraction* A *is called* prevalent *if* $\sup([t]_A) \geq \theta$*, where* $\sup\colon \mathcal{P}(\mathcal{O}) \to [0, 1]$ *is a measure for subsets of* $\mathcal{L}$ *and* $\theta \in [0, 1]$ *is a threshold parameter.*

Note that an ontology, can contain both prevalent and non-prevalent syntactic regularities. Also note, that the existence of prevalent or non-prevalent syntactic regularities is not sufficient to consider an ontology as a whole syntactically regular or irregular. Rather, an ontology is syntactically regular, if it consists *to a large extent* of syntactic regularities. Otherwise, it is irregular.

Definition 8 (k-Regularity). *A (finite) language $\mathcal{L}$ is k-*regular *($k \in \mathbb{N}$) w.r.t. a language abstraction A if there exist $t_1, \ldots, t_k \in \mathcal{L}$ s.t.* $\sup([t_1]_A, \ldots, [t_k]_A) \geq \theta$*, where* $\sup \colon \mathcal{P}(\mathcal{O}) \to [0, 1]$ *is a measure for subsets of $\mathcal{L}$ and $\theta \in [0, 1]$ is a threshold parameter. Otherwise, $\mathcal{O}$ is k-*irregular.

Note that the two notions of prevalent syntactical regularity (cf. Definition 7) and k-regularity (cf. Definition 8) are independent from each other. While both definitions involve a measure function and a threshold parameter, we do not impose a formal relationship between them.

5 Methods

5.1 Research Questions

The notion of syntactic regularities introduced in Section 4 raises a number of research questions for OWL ontologies. Here, we distinguish between two broad categories of such questions. On the one hand, we are interested in the way OWL's language constructors are used and composed in practice. On the other hand, we are interested in identifying reoccurring structural components beyond the use of language constructors that may reveal design decisions or common modelling practices in ontology engineering.

For the purpose of developing a first understanding of syntactic regularities in OWL ontologies, we focus on class expression axioms.[7] This choice is motivated by the observation that OWL ontologies tend to contain large numbers of class expression axioms [6]. Also, we explore notions of syntactic regularity for *sets* of axioms on the basis of the widely-used notion of *class frames* that is used in both practice and theory as evidenced by popular ontology engineering tools and research [2,10,5].

5.2 Experimental Design

We use the following three language abstractions for class expression axioms for our empirical investigation: (i) a ground generalisation over an ontology's class names, property names, and individual names, (ii) a ground generalisation over all constant symbols in OWL, (iii) a selection for class constructors. In the following, we refer to these three language abstractions as signature generalisation (SG), ground generalisation (GG), and class constructor preservation (CCP) respectively.

Using these three language abstractions, we conduct three distinct experiments. For each of these experiments, we distinguish between two experimental conditions: (a) syntactic regularities for *axioms* and (b) syntactic regularities for *sets of axioms*.

[7] `https://www.w3.org/TR/owl2-syntax/#Class_Expression_Axioms`

While there are numerous ways sets of axioms over an ontology can be defined, we use the notion of class frames to induce a family of sets over an ontology. In particular, for a class C occurring in an ontology $\mathcal{O}$, we define its corresponding class frame $CF(\mathsf{C}, \mathcal{O})$ by

$$\begin{aligned} CF(\mathsf{C}, \mathcal{O}) = {} & \{\texttt{SubClassOf}(\mathsf{C}, \mathsf{C}') \in \mathcal{O}\} \cup \\ & \{\texttt{EquivalentClasses}(\mathsf{C}, \mathsf{C}'_1, \ldots, \mathsf{C}'_n) \in \mathcal{O}\} \cup \\ & \{\texttt{DisjointClasses}(\mathsf{C}, \mathsf{C}'_1, \ldots, \mathsf{C}'_n) \in \mathcal{O}\} \cup \\ & \{\texttt{DisjointUnion}(\mathsf{C}, \mathsf{C}'_1, \ldots, \mathsf{C}'_n) \in \mathcal{O}\} \end{aligned}$$

With this, we specify the three aforementioned experiments as follows.

Syntactic Diverseness We determine the syntactic diverseness w.r.t. to a given language abstraction in an ontology by counting the number of syntactic regularities embodied in an ontology. A large number of syntactic regularities suggests that an ontology is syntactically diverse while a small number indicates syntactical homogeneity.

Prevalent Regularities We investigate the extent to which syntactic regularities in ontologies are prevalent. We measure the prevalence of a syntactic regularity with the measuring functions $\sup(x) \mapsto \frac{|x|}{|\Omega|}$ in condition (a) for axioms and $\sup(x) \mapsto \frac{|\{\alpha \in \Omega \mid \exists f \in x \colon \alpha \in f\}|}{|\Omega|}$ in condition (b) for class frames. Here, Ω denotes the set of all class expression axioms.

We decide whether a given syntactic regularity is prevalent or not by using the threshold parameter $\theta = 0.1$. So intuitively, a syntactic regularity is *prevalent* if it accounts for at least 10% of all class expression axioms in a given ontology. We report on the number of prevalent syntactic regularities and give an account of their syntactic properties.

Ontology Coverage with Regularities We analyse an ontology's k-regularity w.r.t. a threshold parameter $\theta = 0.9$ and use the measuring functions and $\sup(x_1, \ldots, x_k) \mapsto \frac{|\bigcup_k x|}{|C|}$ for condition (a), and $\sup(x_1, \ldots, x_k) \mapsto \frac{|\bigcup(\bigcup_k x)|}{|\Omega|}$ for condition (b). For condition (a) we determine the smallest k and in condition (b) we approximate k by iteratively considering the k-most prevalent class frame regularities.

5.3 Ontology Corpus

We work with a recent (June 2020) snapshot of BioPortal created in the same way[8] as described in [7]. The data set of ontologies with their imports closure merged in encompasses a total of 622 ontologies. In experiments, we distinguish between three kinds of ontologies. First, ontologies that consist of atomic axioms only, i.e., `SubClassOf` and `EquivalentClasses` axioms that have only named classes as arguments. Second, ontologies expressible in $\mathcal{EL}^{++}$. And third, ontologies not expressible in $\mathcal{EL}^{++}$. We refer to these three kinds of ontologies as *atomic*, $\mathcal{EL}^{++}$, and *rich* ontologies respectively. Figure 13 shows the size of an ontology's TBox as well as the size of its subset of class expression axioms.

[8] `https://github.com/matentzn/bioportal.download`

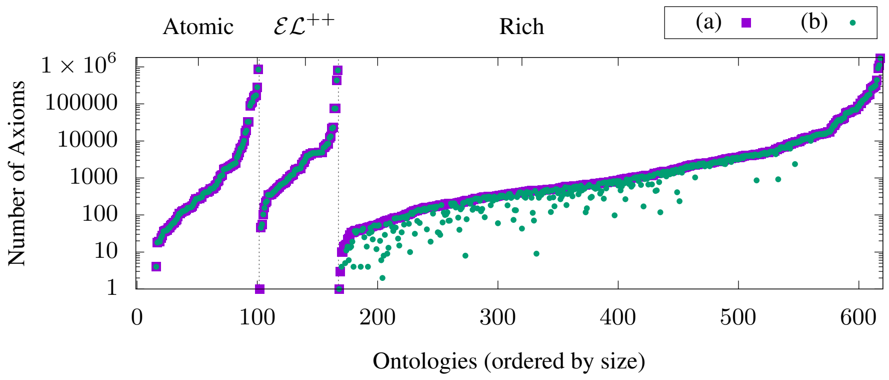

Fig. 13: Number of TBox axioms (a) and Class Expression Axioms (b).

6 Results

We present results for the three experiments as specified in Section 5.2 in distinct subsections. Each subsection is further subdivided according to the two experimental conditions for syntactic regularities for (a) axioms and (b) sets of axioms. In each case, we report on findings w.r.t. to the three language abstractions (i)–(iii).

6.1 Experiment 1: Syntactic Diverseness

(a) Axioms Figure 14 shows the (absolute) counts of syntactic regularities for axioms in BioPortal ontologies. We observe that the number of regularities (positively) correlates with both the used language profile (atomic, $\mathcal{EL}^{++}$, and rich) as well as the size of an ontology. However, most ontologies contain at most 100 syntactic regularities for all three language abstractions. In fact, it appears that the language abstractions, SG, GG, and CCP often coincide in terms of the number of syntactic regularities. This happens, for example, when an ontology does not contain axioms that the three language abstractions treat differently or if syntactic features that are treated differently are used in a homogeneous manner.

Yet, we also observe ontologies with large difference between the number of syntactic regularities for different language abstractions. For example, the Orphanet Rare Disease Ontology at index 166, contains a large number of syntactic regularities w.r.t. SG. This is due to the use of literal value restrictions, e.g., `DataHasValue`(Orphanet_C029 "1.9"^^xsd:float). Since SG does not abstract over literals, axioms with different literals, e.g., different (float) numbers, belong to different syntactic regularities. The number of regularities w.r.t. GG is comparatively small because GG abstracts over differences between literal values. For example, the literal value restriction for Orphanet_C029 and `DataHasValue`(Orphanet_C024 "50.0"^^xsd:float) belong to different syntactic regularities w.r.t SG but to the same w.r.t. GG.

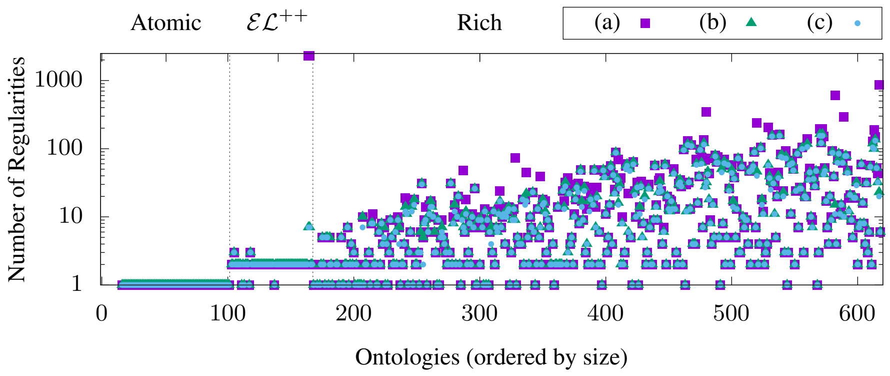

Fig. 14: Syntactic Diverseness for axioms w.r.t. (a) SG, (b) GG, and (c) CCP.

(b) Class Frames Figure 15 shows the (absolute) counts of syntactic regularities for class frames in BioPortal ontologies. We make the same observations as for syntactic regularities for axioms. Namely, the number of regularities correlates positively with both an ontology's language profile as well as its size. Also, the three language abstractions SG, GG, and CCP generally give rise to the same number of syntactic regularities. However, we need to point out the overall increased number of regularities compared to the case of (a) axioms. This is due to the fact that axioms of the same syntactic regularity can be combined in various ways to define different syntactic regularities for class frames. For example, consider the ontology

$$\mathcal{O} = \{\texttt{SubClassOf}(\mathsf{A}, \mathsf{B}), \texttt{SubClassOf}(\mathsf{A}, \mathsf{B}), \texttt{SubClassOf}(\mathsf{C}, \mathsf{D})\}.$$

There is only one syntactic regularity for axioms (w.r.t. GG) but two for class frames.

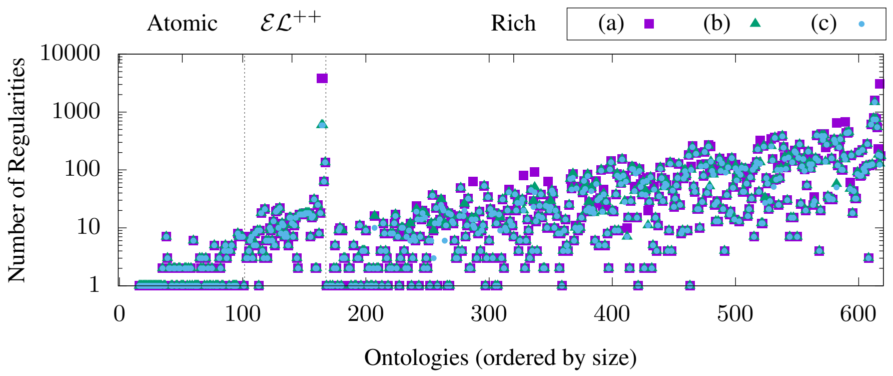

Fig. 15: Syntactic Diverseness for class frames w.r.t. (a) SG, (b) GG, and (c) CCP.

6.2 Experiment 2: Prevalent Regularities

(a) Axioms In light of only one or two syntactic regularities for almost all atomic and $\mathcal{EL}^{++}$ ontologies (see Figure 14), it comes to no surprise that these regularities are also often prevalent, i.e., they make up at least 10% of the class expression axioms in an ontology. There is only one $\mathcal{EL}^{++}$ with more than two prevalent regularities, namely the Orphanet Rare Disease Ontology with four regularities w.r.t. GG and CCP (recall that SG gives rise to a large number of syntactic regularities due to literal value restrictions. However, these regularities contain relatively few axioms).

Similarly, the three language abstractions give rise to at most two prevalent regularities for rich ontologies as well. This is also to be expected as (atomic) axioms from the class hierarchy often make up large a proportion of an ontology's TBox [4]. Yet, about 15% of rich ontologies (68 out of 451) exhibit more than two prevalent regularities w.r.t. at least one of the three used language abstractions; but only about 4% (17 out of 451) exhibit more than three prevalent regularities.

Otherwise, there is no ontology with more than four prevalent regularities with the exception of only one, namely the "Neomark Oral Cancer Ontology (version 3)." Note that this ontology only contains 236 class expression axioms. So any syntactic regularity with more than 23 axioms is considered prevalent. In fact, rich ontologies with more than three prevalent regularities tend to have a "small" TBox. In particular, there are only four ontologies with at least 1000 TBox axioms that exhibit more than 3 prevalent regularities.

(b) Class Frames For class frames, we find slightly more prevalent syntactic regularities compared to regularities for (single) axioms. More than 90% ontologies exhibit at most three prevalent regularities for all language abstractions (573 out of 619). Yet, there are a few ontologies in which up to 8 regularities are prevalent. This is in line with the overall larger number of syntactic regularities for class frames (cf. Section 6.1). However, it is important to keep in mind that our notion of class frames does not induce a partition of an ontology; neither do syntactic regularities for class frames. In fact, both `EquivalentClasses` and `DisjointClasses` axioms can be part of multiple class frames associated with different classes. This can (in theory) lead to a large number of prevalent regularities. Consider the following example ontology:

$$\begin{aligned}\mathcal{O} = \{\ & \texttt{DisjointClasses}(\mathsf{A},\mathsf{B}),\ \texttt{DisjointClasses}(\mathsf{B},\mathsf{C}),\\ & \texttt{DisjointClasses}(\mathsf{A},\mathsf{C}),\ \texttt{DisjointClasses}(\mathsf{B},\mathsf{D}),\\ & \texttt{DisjointClasses}(\mathsf{A},\mathsf{D}),\ \texttt{DisjointClasses}(\mathsf{C},\mathsf{D}),\\ \\ & \texttt{SubClassOf}(\mathsf{A},\exists \mathsf{R}.\mathsf{X}) \qquad \texttt{SubClassOf}(\mathsf{B},\mathsf{X}\sqcap\mathsf{Y}),\\ & \texttt{SubClassOf}(\mathsf{C},\forall \mathsf{R}.\mathsf{X}) \qquad \texttt{SubClassOf}(\mathsf{D},\mathsf{X}\sqcup\mathsf{Y})\}\end{aligned}$$

Here, the four classes A, B, C, and D all give rise to a class frame involving four axioms. Since all class frames belong to different regularities (due to the `SubClassOf` axioms) and $\mathcal{O}$ only contains 10 axioms, all four regularities would be considered prevalent w.r.t. a threshold of 40%. However, if the class frames would induce a partition of $\mathcal{O}$, then there should be at most two syntactic regularities that cover 40% of class expressions in $\mathcal{O}$.

6.3 Experiment 3: Ontology Coverage

(a) Axioms Most ontologies contain only one or two prevalent regularities (cf. Section 6.1). Yet, the total number of syntactic regularities is larger than 10 for many rich ontologies and even larger 100 in some cases (cf. Section 6.1). In Figure 16 we show the number of syntactic regularities that need to be merged to cover 90% of class expression axioms in an ontology. We observe that two regularities are often sufficient. Furthermore, we note that cases in which more than 10 regularities are needed are rare.

This suggests that ontologies, for the most part, are built on the basis of (syntactically) homogeneous set of axioms. As argued before, this observation can (to large extents) be attributed to the predominant role of atomic subsumption axioms that are used to represent an ontology's class hierarchy. Also, it is important to keep in mind that most entities in OWL are represented by a *combination* of a set of axioms. Therefore, it is not warranted to assume an ontology to follow an overall homogeneous design only because it is predominantly built on the basis of axioms of a certain form.

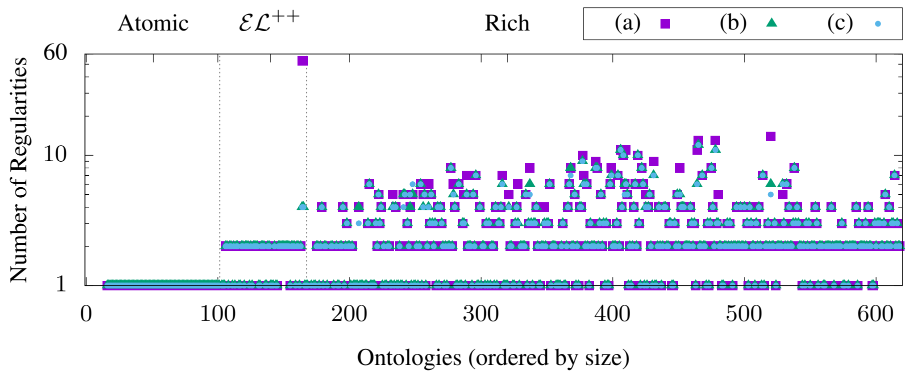

Fig. 16: Number of axiom regularities needed to cover 90% of an ontology's class expression axioms w.r.t. (a) SG, (b) GG, and (c) CCP.

To gain a better understanding of an ontology's composition w.r.t. syntactic regularities, we inspect what proportion of an ontology's class expression axioms are covered by the three most prevalent regularities w.r.t. SG. We only discuss SG because GG and CCP abstract over a superset of syntactical features compared to SG and thus give rise to fewer regularities (of bigger size). While 16 already shows that many ontologies' class expression axioms can be covered to 90% by only three regularities, Figure 17 shows that 50% of class expression axioms can be covered by the three most prevalent regularities in all ontologies. There are only 17 ontologies whose class expression axioms are covered to less than 70% by the three most prevalent regularities.

Note that some ontologies contain no class expression axioms. Such ontologies are represented in Figure 17 with a zero percentage and we excluded them for the above counts as it is a question of taste whether an empty set is covered to 0% or 100%.

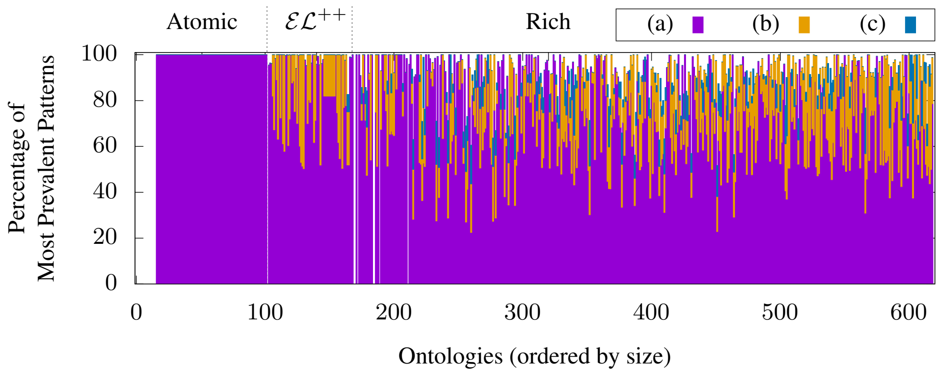

Fig. 17: Three most prevalent syntactic regularities w.r.t. SG: (a) depicts the most prevalent, (b) the second most prevalent, and (c) the third most prevalent.

(b) Class Frames For class frames, we find that class expression axioms in atomic and $\mathcal{EL}^{++}$ ontologies can also be covered to 90% by just 10 regularities (with a few exceptions). However, for rich ontologies we observe that the number of needed regularities increases with the size of an ontology and often exceeds 10 considerably.

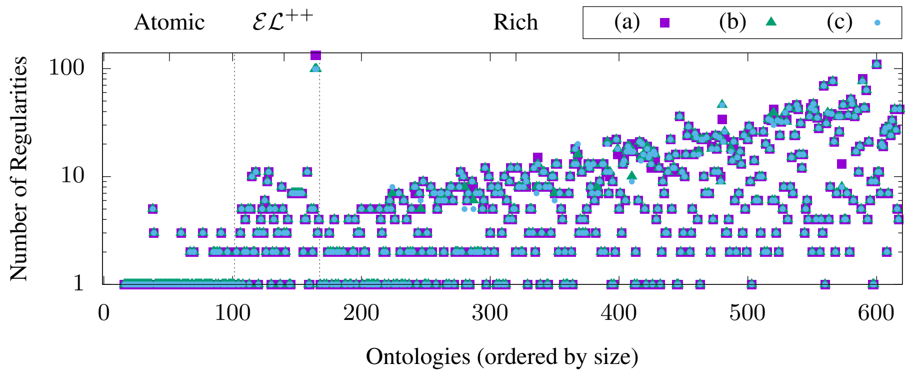

Fig. 18: Number of class frame regularities needed to cover 90% of an ontology's class expression axioms w.r.t. (a) SG, (b) GG, and (c) CCP.

As before, we inspect the three most prevalent syntactic regularities w.r.t. SG and determine what proportion of an ontology's class expression axioms they already cover. Figure 19 reveals that the three most prevalent regularities for class frames are not as prominent as for axioms. While there are ontologies for which the three most prevalent regularities are sufficient to cover 80% of their class expression axioms, there are equally as many ontologies for which this does not hold.

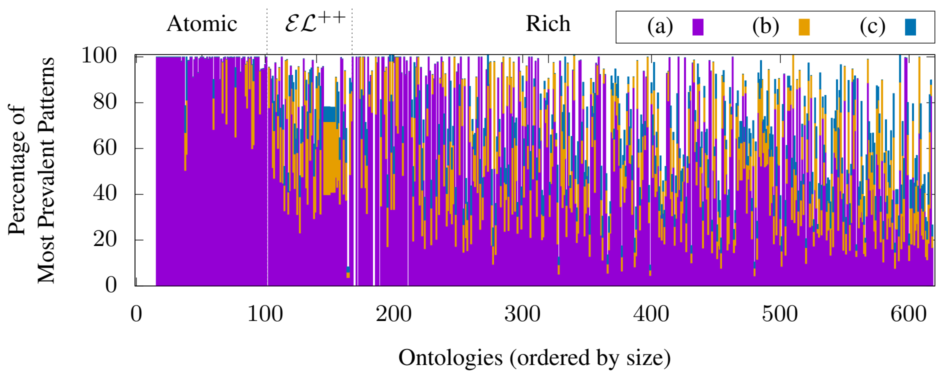

Fig. 19: Three most prevalent syntactic regularities w.r.t. SG. (a) depicts the most prevalent, (b) the second most prevalent, and (c) the third most prevalent.

7 Related Work

A range of approaches for discovering syntax-based regularities in OWL ontologies have been proposed. First, agglomerative clustering has been used to identify commonalities between similar entities [8]. The used notion of a *replacement function* can be interpreted as a language abstraction. However, its use in the similarity distance measure effectively reduces a set of axioms to a set of *types* of axioms. While the resulting loss of information can be seen as a form of abstraction, it is not directly comparable to the language abstractions motivated in this work.

Second, frequent subtree mining over OWL axioms has been motivated by interpreting them as (syntax) trees [5]. Furthermore, by using frequent itemset mining over identified regularities for axioms, regularities for sets of axioms are motivated. While frequent subtree mining aims to identify frequent structures, syntactic regularities based on language abstractions *do not*. Language abstractions are primarily concerned with syntactic properties and the corresponding notion for regularities is independent from any notion of frequency. The qualification of regularities in terms of being prevalent, i.e., capturing frequently occurring syntactic structures, is a separate level of analysis.

Third, lexical naming conventions for OWL classes in combination with structural relations between them in an ontology's class hierarchy have been suggested to provide useful information w.r.t. an ontology's underlying design [12].

Besides, approaches for discovering regularities from a given ontology alone, there also exists target-oriented approaches that aim to detect some predefined notion of patterns [3,9,11]. Otherwise, there are surveys of ontologies discussing syntactic properties such as the prevalence of, for example, logical constructors [6,13].

8 Conclusion

We have proposed a formal framework for identifying and characterising syntactic regularities in ontologies. We used this framework to survey a large corpus of actively maintained ontologies and found prevalent regularities that may prove to be useful in terms of ontology comprehension and maintenance. While the used language abstractions in our empirical survey are somewhat coarse-grained, they can easily be refined or otherwise modified as needed. Overall, this work is motivated by the need for data-driven methods to extract and characterise common or recurring modelling practices in published ontologies. The hope is that such an automated information extraction will help with the development of high-quality design patterns both in theory and practice.

References

1. Comon, H.: Tree automata techniques and applications. http://www. grappa. univ-lille3. fr/-tata (1997)
2. Horridge, M., Patel-Schneider, P.F.: Manchester syntax for OWL 1.1. In: OWLED (Spring). CEUR Workshop Proceedings, vol. 496. CEUR-WS.org (2008)
3. Kindermann, C., Parsia, B., Sattler, U.: Detecting influences of ontology design patterns in biomedical ontologies. In: ISWC (1). Lecture Notes in Computer Science, vol. 11778, pp. 311–328. Springer (2019)
4. Kindermann, C., Parsia, B., Sattler, U.: Prevalence and effects of class hierarchy precompilation in biomedical ontologies. In: International Semantic Web Conference (1) (2020)
5. Lawrynowicz, A., Potoniec, J., Robaczyk, M., Tudorache, T.: Discovery of emerging design patterns in ontologies using tree mining. Semantic Web **9**(4), 517–544 (2018)
6. Matentzoglu, N., Bail, S., Parsia, B.: A snapshot of the OWL web. In: International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 8218, pp. 331–346. Springer (2013)
7. Matentzoglu, N., Parsia, B.: Bioportal snapshot 30.03.2017 (Mar 2017). https://doi.org/10.5281/zenodo.439510, `https://doi.org/10.5281/zenodo.439510`
8. Mikroyannidi, E., Manaf, N.A.A., Iannone, L., Stevens, R.: Analysing syntactic regularities in ontologies. In: Klinov, P., Horridge, M. (eds.) Proceedings of OWL: Experiences and Directions Workshop 2012, Heraklion, Crete, Greece, May 27-28, 2012. CEUR Workshop Proceedings, vol. 849. CEUR-WS.org (2012), `http://ceur-ws.org/Vol-849/paper_11.pdf`
9. Mortensen, J., Horridge, M., Musen, M.A., Noy, N.F.: Modest use of ontology design patterns in a repository of biomedical ontologies. In: WOP. CEUR Workshop Proceedings, vol. 929. CEUR-WS.org (2012)
10. Noy, N.F., Musen, M.A., Jr., J.L.V.M., Rosse, C.: Pushing the envelope: challenges in a frame-based representation of human anatomy. Data Knowl. Eng. **48**(3), 335–359 (2004)
11. Sváb-Zamazal, O., Scharffe, F., Svátek, V.: Preliminary results of logical ontology pattern detection using SPARQL and lexical heuristics. In: WOP. CEUR Workshop Proceedings, vol. 516. CEUR-WS.org (2009)
12. Sváb-Zamazal, O., Svátek, V.: Analysing ontological structures through name pattern tracking. In: EKAW. Lecture Notes in Computer Science, vol. 5268, pp. 213–228. Springer (2008)
13. Wang, T.D., Parsia, B., Hendler, J.A.: A survey of the web ontology landscape. In: International Semantic Web Conference. Lecture Notes in Computer Science, vol. 4273, pp. 682–694. Springer (2006)