Concrete Names for Complex Expressions in Ontologies: A Case Study on SNOMED CT

Christian Kindermann*, Martin Georg Skjæveland

University of Oslo, Problemveien 7, 0315 Oslo, Norway

Abstract

Giving an entity in a knowledge graph a dedicated name raises the question of whether this name is used consistently in the knowledge graph or not. In this paper, we investigate the way complex concepts, e.g., medical diseases, are represented and used in knowledge graphs. In particular, we wonder whether knowledge graphs include definitions of concise but informative names for more complex concepts that can be used in lieu of more explicit but lengthy representations.

For this purpose, we formalise the idea of introducing dedicated names for complex expressions in OWL. We use this formalisation to conduct a case study on SNOMED CT. We analyse how often names for complex concepts are introduced, how often these names are used throughout an ontology, and whether these names are used consistently whenever they can be used. We find that many complex concepts in SNOMED CT are associated with a dedicated name and that such names are mostly used in a consistent fashion throughout the ontology. However, there are a few exceptions and there are also cases of complex concepts that are not associated with a dedicated name.

1. Introduction

The representation of an entity in a knowledge graph often involves statements about the entity's characteristics. In the case these characteristics can be associated with a concrete idea, it may be advantageous to make this explicit by *naming* the idea rather than describing it. For example, consider the description of an animal with large ears, a trunk, tusks, and a weight of several tons. The animal in question is of course an *elephant*.

A name, e.g., elephant, provides an informative and concise reference that can be used in lieu of a possibly complex description. This can be useful in the context of summarising information in a knowledge graph. Entities that all share the same characteristics (of say an elephant) can either be represented by (i) repeatedly enumerating all characteristics for all entities individually or, alternatively, by (ii) defining the notion of an elephant in terms of said characteristics, and then declaring entities as elephants using a reference to the definition of what it means to be an elephant.

To make this argument more concrete, consider a set of entities $e_1, \ldots, e_n$ and a set of characteristics $c_1, \ldots, c_m$ for elephants. Using approach (i), a knowledge graph written in say RDF would require (at least) n times m many triples of the form (e_i, p, c_j) for $1 \leq i \leq n$

Joint proceedings of ISWC2022 Workshops: the International Workshop on Artificial Intelligence Technologies for Legal Documents (AI4LEGAL) and the International Workshop on Knowledge Graph Summarization (KGSum), 2022

*Corresponding author.

chrikin@ifi.uio.no (C. Kindermann); martige@ifi.uio.no (M. G. Skjæveland)

and $1 \leq j \leq m$ where p is some property that associates entity e_i with characteristic c_j. On the other hand, using approach (ii), only n triples of the form (e_i, `rdf:type`, Elephant) are required, plus (approximately) m triples to define the meaning of Elephant in terms of the m characteristics.

The *meaning* of a name, e.g., Elephant, in a knowledge graph can be specified via the semantics of a knowledge representation language, e.g., OWL [1]. Once the meaning of a name is formally specified in a knowledge graph, it can be used throughout the knowledge graph to represent its associated information. However, even though it seems plausible to always use concise and informative names where this is possible, it has been observed that this is not always done in published knowledge graphs. We repeat a concrete example taken from the Galen ontology that has originally been presented by Nikitina and Koopmann [2]. Here, the medical concept Clotting is represented as follows:

$$\begin{aligned}\mathsf{Clotting} \equiv \exists\, \mathsf{actsSpecificallyOn}.(&\mathsf{Blood} \sqcap \\ &\exists\, \mathsf{hasPhysicalState}.(\mathsf{PhysicalState} \sqcap \\ &\qquad \exists\, \mathsf{hasState.Liquid})) \sqcap \\ &\exists\, \mathsf{hasOutcome.SolidBlood})\end{aligned}$$

This axiom is arguably complex due to both its size and the nesting of expressions. However, Galen also contains the following axioms:

$$\begin{aligned}\mathsf{LiquidBlood} &\equiv \mathsf{Blood} \sqcap \exists\, \mathsf{hasPhysicalState.LiquidState} \\ \mathsf{LiquidState} &\equiv \mathsf{PhysicalState} \sqcap \exists\, \mathsf{hasState.Liquid}\end{aligned}$$

Given these equivalences, the named concept LiquidBlood can be used to simplify the representation of Clotting to

$$\begin{aligned}\mathsf{Clotting} \equiv &\exists\, \mathsf{actsSpecificallyOn.LiquidBlood} \sqcap \\ &\exists\, \mathsf{hasOutcome.SolidBlood}\end{aligned}$$

This observation provides the motivation for investigating whether names for more complex expressions are in fact used in their stead. The contributions are as follows: (i) we propose an approach for identifying definitions of various kinds for named classes in ontologies, (ii) we develop techniques for quantifying the use and lack of reuse of such named classes, and (iii) we use these techniques to conduct a case study on SNOMED CT [3] as an example ontology in which more complex concepts are automatically constructed based on a compositional grammar.

2. Preliminaries

We assume the reader to be familiar with OWL [1] and only fix some terminology. Let N_C, N_I, and N_P be sets of *class names*, *individual names*, and *property names*. A *class* is either a class name or a *complex class* built using OWL class constructors. We will use $\top$ and $\bot$ to denote `owl:Thing` and `owl:Nothing` respectively. We use both OWL Functional Style Syntax [4] and Manchester Syntax [5] to write OWL axioms. An ontology is a set of axioms and we write

$\mathcal{O} \models \alpha$ to denote that the ontology $\mathcal{O}$ entails the axiom α. An axiom α is *explicit* in $\mathcal{O}$ if $\alpha \in \mathcal{O}$, and *implicit* if $\alpha \notin \mathcal{O}$ but $\mathcal{O} \models \alpha$. An OWL expression e *occurs* in $\mathcal{O}$ if e is used as a subexpression within an explicit axiom in $\mathcal{O}$.

3. Abbreviations in Ontologies

3.1. Abbreviations and Synonyms

The Oxford English Dictionary defines the word *abbreviation* to denote "[t]he result of shortening something; an abbreviated or condensed form, esp. of a text; a summary, an abridgement" [6]. So, we define an abbreviation for a complex OWL expression as an equivalent named class.

Definition 1 (Abbreviation). *A named class* A *is an* abbreviation *for a complex class expression* C *in an ontology* $\mathcal{O}$ *if* $\mathcal{O} \models \mathit{EquivalentClasses}(\mathsf{A}, \mathsf{C})$. *The axiom* $\mathit{EquivalentClasses}(\mathsf{A}, \mathsf{C})$ *is called the* definition *of* A.

A complex OWL expression can be equivalent to more than just one named class. We refer to equivalent named classes as *synonyms*.[1] Please note that synonyms are not necessarily abbreviations. However, a synonym for an abbreviation is also an abbreviation (due to transitivity of $\mathit{EquivalentClasses}$).

Definition 2 (Synonym). *A* synonym *for a named class* N *in an ontology* $\mathcal{O}$ *is a named class* S *s.t.* $\mathcal{O} \models \mathit{EquivalentClasses}(\mathsf{S}, \mathsf{N})$. *The axiom* $\mathit{EquivalentClasses}(\mathsf{S}, \mathsf{N})$ *is called the* definition *of* S.

Both abbreviations and synonyms are notions based on entailment, i.e., an *EquivalentClasses* axiom with exactly two arguments. However, OWL specifies *EquivalentClasses* as an n-ary constructor. So, for the purpose of analysing how abbreviations and synonyms are specified in ontologies, we introduce a notion that accounts for all syntactic variants of $\mathit{EquivalentClasses}$.

Definition 3 (Definition Types). *Let* A *be an abbreviation,* $\mathsf{S}_1, \ldots, \mathsf{S}_m$ *synonyms, and* $\mathsf{C}_1, \ldots, \mathsf{C}_n$ *complex class expressions. Then, an axiom* α *is a*

- simple definition *of* A, *if* $\alpha = \mathit{EquivalentClasses}(\mathsf{A}, \mathsf{C}_i)$,
- simple definition *of* S_i, *if* $\alpha = \mathit{EquivalentClasses}(\mathsf{A}, \mathsf{S}_i)$,
- ambiguous definition *of* A, *if* $\alpha = \mathit{EquivalentClasses}(\mathsf{A}, \mathsf{C}_1, \ldots, \mathsf{C}_n)$,
- enumerative definition *for* $\mathsf{S}_1, \ldots, \mathsf{S}_m$, *if* $\alpha = \mathit{EquivalentClasses}(\mathsf{S}_1, \ldots, \mathsf{S}_m)$,
- compound definition *for* $\mathsf{S}_1, \ldots, \mathsf{S}_m$, *if* $\alpha = \mathit{EquivalentClasses}(\mathsf{S}_1, \ldots, \mathsf{S}_m, \mathsf{C}_1, \ldots, \mathsf{C}_n)$.

With this notion of definition types, we can quantify how abbreviations and synonyms are specified in an ontology, i.e., we can count both explicit as well as implicit $\mathit{EquivalentClasses}$ in an ontology. Since the extraction of finite sets of entailments is a non-trivial matter [8], we will discuss how we determine and count implicit abbreviations and synonyms in more detail in Section 3.3. However, before we do so, we discuss the more obvious question about how abbreviations and synonyms are *used* in an ontology.

[1] The Oxford English Dictionary defines the word *synonym* to denote "Strictly, a word having the same sense as another (in the same language); [. . .]" [7].

α_1 = SpicyPizza EquivalentTo Pizza and
(hasTopping some (PizzaTopping and (hasSpiciness some Hot)))
α_2 = SpicyTopping EquivalentTo PizzaTopping and (hasSpiciness some Hot)
α_3 = SpicyTopping EquivalentTo HotTopping
α_4 = DiavolaPizza SubClassOf SpicyPizza
α_5 = DiavolaPizza SubClassOf Pizza and hasCountryOfOrigin value Italy
α_6 = NapoletanaPizza SubClassOf Pizza and hasCountryOfOrigin value Italy

Figure 1: Example of abbreviations and synonyms in a sample ontology. The named class SpicyPizza is an abbreviation. The classes SpicyTopping and HotTopping are synonyms.

3.2. Using Abbreviations in Ontologies

Consider the example ontology $\mathcal{O}_{Ex}$ shown in Figure 1. Here, the abbreviation SpicyPizza is specified via a simple definition in axiom α_1 and occurs on the right-hand side of α_4. So, we say an abbreviation is *used* if it occurs in an OWL axiom that is not its definition.

Definition 4 (Abbreviation Use). *Let* A *be an abbreviation in an ontology* $\mathcal{O}$*. If* A *occurs in an axiom* $\alpha \in \mathcal{O}$ *and* α *is not a definition for* A *(according to Definition 3), then* A *is* used *in* α*.*

In addition to the use of an abbreviation, we can also determine if an abbreviation is *not used* even though its use would be possible.

Definition 5 (Possible Abbreviation Use). *Let* $EquivalentClasses(\mathsf{A}, C)$ *be the definition of an abbreviation* A *in an ontology* $\mathcal{O}$*. If* C *occurs in an axiom* $\alpha \in \mathcal{O}$ *and* α *is not a definition for* A *(according to Definition 3), then there is a* possible use *for* A *in* α*.*

Consider axiom $\alpha_1 \in \mathcal{O}_{Ex}$. Here, the abbreviation SpicyTopping or its synonym HotTopping have possible uses since the complex OWL expression PizzaTopping and (hasSpiciness some Hot) could be replaced by either SpicyTopping or HotTopping.

Note that the definition of a possible use for an abbreviation is dependent on the existence of an abbreviation in an ontology. However, in cases where no abbreviation is available, one may wonder when it makes sense to introduce an abbreviation for a reoccurring complex expression. A high number of occurrences of a complex class in an ontology may be used as an indicator for the identification of potential candidates for a new abbreviation.

Definition 6 (Abbreviation Candidate). *Let* C *be a complex expression in an ontology* $\mathcal{O}$ *s.t. there is no abbreviation for* C*. If* C *occurs at least* T *times in* $\mathcal{O}$*, then* C *is a* abbreviation candidate *w.r.t. threshold* T*.*

With the notions of an abbreviation's use (Definition 4), possible use (Definition 5), and abbreviation candidate (Definition 6), we can quantify the impact of abbreviations in an ontology. First, however, one needs to identify all abbreviations in an ontology. So, in the next section, we specify a procedure for determining all explicit as well as implicit abbreviations defined in an ontology.

3.3. Determining Abbreviations

Explicit definitions for abbreviations (according to Definition 3) can be determined in a straightforward manner by checking the syntactic shape of all axioms in a given ontology. Similarly, implicit definitions can be determined in a straightforward manner by checking $\mathcal{O} \models \mathit{EquivalentClasses}(\mathsf{A}, \mathsf{C})$ for all pairs of named classes and complex classes occurring in an ontology. However, this is impractical for large ontologies with a large number of both named and complex classes.

Instead, we propose to determine implicit abbreviations by building on highly optimised implementations of the standard reasoning service *classification*, i.e., the computation of all entailed $\mathit{SubClassOf}$ and $\mathit{EquivalentClasses}$ axioms between named classes in a given ontology [9, 10, 11]. We will refer to this set as the inferred class hierarchy (ICH).

The idea is to first introduce abbreviations for all complex class expressions that occur in a given ontology, then to compute its ICH, and finally to read off all implicit abbreviations from the ICH. So, more formally, for a given ontology $\mathcal{O}$, we create the ontology

$$\mathcal{O}_A = \mathcal{O} \cup \{\mathit{EquivalentClasses}(\mathsf{A}_i, \mathsf{C}_i) \mid \mathsf{C}_i \text{ occurs in } \mathcal{O}, \mathsf{A}_i \text{ does not occur in } \mathcal{O}\}$$

and compute $\mathsf{ICH}(\mathcal{O}_A)$. Since the ICH captures all $\mathit{SubClassOf}$ and $\mathit{EquivalentClasses}$ relationships between named classes in an ontology, it is straightforward to identify all named classes in $\mathcal{O}$ that are equivalent to a newly introduced abbreviation A_i in $\mathcal{O}_A$. We will present a worked example of this approach and will discuss a number of technical details that are not mentioned in the high-level description outlined above.

Consider $\mathcal{O}_{Ex}$ shown in Figure 1. The ontology consists of six axioms in which the six complex class expressions $\mathsf{C}_1, \ldots, \mathsf{C}_6$, as shown in Figure 2a, occur. So, given these six class expressions, we construct the ontology

$$\mathcal{O}_{Ex}^{A} = \mathcal{O}_{Ex} \cup \{\mathit{EquivalentClasses}(\mathsf{A}_1, \mathsf{C}_1), \ldots, \mathit{EquivalentClasses}(\mathsf{A}_6, \mathsf{C}_6)\}$$

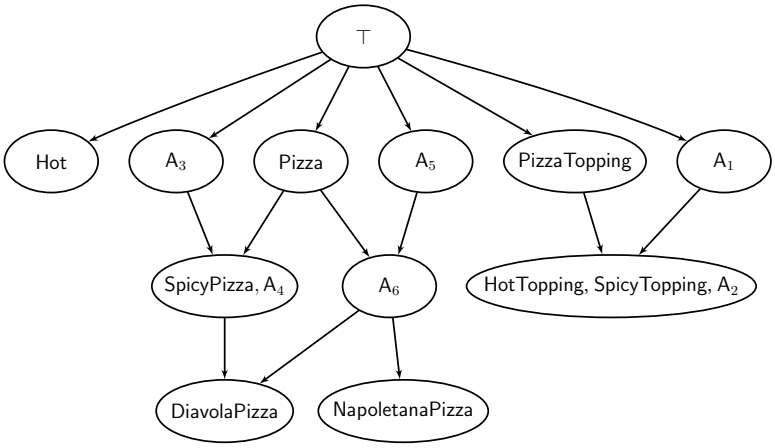
where $\mathsf{A}_1, \ldots, \mathsf{A}_6$ are newly created class names that do not occur in $\mathcal{O}_{Ex}$. We then classify $\mathcal{O}_{Ex}^{A}$ and represent the resulting ICH by a transitive reduct (w.r.t. the $\mathit{SubClassOf}$ relation) in which equivalent classes are grouped into collections. More formally, let $[\mathsf{C}]_{\mathcal{O}}$ denote the set $\{\mathsf{C} \mid \mathcal{O} \models \mathit{EquivalentClasses}(\mathsf{C}, \mathsf{C}_i)\}$. By abuse of notation let $\mathit{SubClassOf}([\mathsf{X}]_{\mathcal{O}}, [\mathsf{Y}]_{\mathcal{O}})$ denote the set of axioms $\{\mathit{SubClassOf}(\mathsf{X}_i, \mathsf{Y}_j) \mid \mathsf{X}_i \in [\mathsf{X}]_{\mathcal{O}}, \mathsf{Y}_j \in [\mathsf{Y}]_{\mathcal{O}}\}$. With this, the transitive reduct of an ontology's ICH is defined as the set of $\mathit{SubClassOf}$ axioms between equivalent classes, i.e.,

$$\mathsf{TR}(\mathcal{O}) = \{\mathit{SubClassOf}([\mathsf{X}]_{\mathcal{O}}, [\mathsf{Y}]_{\mathcal{O}}) \mid \mathcal{O} \models \mathit{SubClassOf}([\mathsf{X}]_{\mathcal{O}}, [\mathsf{Y}]_{\mathcal{O}})\}.$$

Figure 2 shows the transitive reduct for the ICH of $\mathcal{O}_{Ex}^{A}$ (omitting $\bot$ for presentational purposes). With this representation of the ICH, it is straightforward to read off all abbreviations in $\mathcal{O}_{Ex}$. For example, SpicyPizza is equivalent to A_4, which in turn is equivalent to C_4. So, SpicyPizza is an abbreviation for C_4 in $\mathcal{O}_{Ex}$. Likewise, it is straightforward to identify complex class expressions without abbreviations.

C_1 = hasSpiciness some Hot
C_2 = PizzaTopping and (hasSpiciness some Hot)
C_3 = hasTopping some (PizzaTopping and (hasSpiciness some Hot))
C_4 = Pizza and (hasTopping some (PizzaTopping and (hasSpiciness some Hot)))
C_5 = hasCountryOfOrigin value Italy
C_6 = Pizza and hasCountryOfOrigin value Italy

(a) Complex class expressions in $\mathcal{O}_{Ex}$.



(b) Visualisation of TR($\mathcal{O}^{A}_{Ex}$) without $\bot$.

Figure 2: Determining implicit abbreviations in $\mathcal{O}_{Ex}$ via the inferred class hierarchy of $\mathcal{O}^{A}_{Ex}$.

4. Methods

4.1. Research Questions

The subject of abbreviations in knowledge bases raises a number of research questions. We distinguish between four broad categories of such questions. The first category is about the *prevalence* of abbreviations in ontologies; both in terms of their definition (cf. Section 3.1) as well as their use (cf. Section 3.2). The second category revolves around how abbreviations, in the sense of concise but informative names for complex expressions, can be *generated and used*. The third category deals with the *claimed benefits* of abbreviations regarding improved ontology comprehension, maintenance, and use in practice. And, finally, the fourth category is concerned with the *potential impact* of abbreviations in the wider context of ontology engineering. For example, how do abbreviations affect common ontology engineering services, e.g., ontology alignment, visualisation, and reasoning only to name a few.

To develop a first understanding of abbreviations in ontologies, we focus on the first category in this work. In particular, we investigate the following questions:

(Q1) How many abbreviations and synonyms are *defined* in ontologies?
(Q2) How often are defined abbreviations and synonyms *used*?

(Q3) How often is an abbreviation *not used* despite being defined?

We also shed some light on the search space of potential candidates (cf. Definition 6) for abbreviations. For this purpose, we investigate the following questions:

(Q4) How often do complex expressions *reoccur* in ontologies?
(Q5) How *large* are such reoccurring complex expressions?
(Q6) What is the *nesting depth* of reoccurring complex expressions?

4.2. Study Design

The notion of an abbreviation is based on entailment (cf. Definition 1). However, an OWL ontology is published as a computational artefact in a concrete syntax. Furthermore, the syntactic structure of an ontology cannot, in general, be assumed to be arbitrary. So, we distinguish between explicit and implicit axioms in the design of our empirical investigation. This investigation consists of four distinct inquiries that we use to analyse the latest international release of SNOMED CT (June 2022). These inquiries concern the (i) definition of abbreviations and synonyms, (ii) their use, (iii) the reoccurrence of complex class expressions, and (iv) the size and nesting depth of such complex class expressions. We give a brief description of each of these inquiries in the following:

1. **Number of Abbreviations and Synonyms**: We determine how many abbreviations and synonyms are defined in SNOMED CT (according to Definition 3). We count explicit and implicit definitions separately following the approach described in Section 3.3. Furthermore, we compare the number of abbreviations and synonyms to the number of named classes in SNOMED CT.
2. **Prevalence of Abbreviation Use**: We determine to what extent abbreviations are used (cf. Definition 4) and to what extent they are *not used* even though this would be possible (cf. Definition 5). We also determine to what extent different synonyms for abbreviations are used.
3. **Reoccurrence of Complex Expression**: We determine how often complex expressions reoccur in SNOMED CT, i.e., how often a given complex class expressions occurs more than once. Furthermore, we analyse to what extent such class expressions are associated with abbreviations or not, i.e., how many complex class expressions are potential abbreviation candidates.
4. **Size and Nesting Depth of Complex Class Expressions**: We determine the size and nesting depth of complex class expressions, where the *size* and *depth* of an expression $expr$ are defined respectively as $size(expr) = 1$ and $depth(expr) = 0$ if $expr$ is atomic, and $size(expr) = 1 + \sum_{i=1}^{n} size(arg_i)$ as well as $depth(expr) = 1 + \max_{1 \leq i \leq n} depth(arg_i)$ if $expr = C(arg_1, \ldots, arg_n)$.

4.3. Materials

We use the international edition of SNOMED CT (June 2022) downloaded via the Unified Medical Language System Terminology Service (http://owlapi.sourceforge.net/). and use the Snomed

Table 1
Number of abbreviations and their uses in SNOMED CT.

Condition	0	1	2–4	5–10	11–20	21–50	51–100	> 100
Explicit	119 076	4 994	3 005	986	285	103	23	10
Implicit	0	0	2 072	975	287	108	16	14

Table 2
Number of complex class expressions associated with a possible abbreviation use in SNOMED CT.

Condition	0	1	2–4	5–10	11–20	21–50	51–100	> 100
Explicit	124 368	2 603	1 210	245	34	18	2	2
Implicit	0	3 552	1 945	308	42	4	1	1

OWL Toolkit (https://github.com/IHTSDO/snomed-owl-toolkit) to convert its release format (RF2) to OWL. All reasoning tasks are performed with ELK version 4.3 (https://github.com/liveontologies/elk-reasoner). The empirical investigation is orchestrated using the OWL API version 5.1.15 (http://owlapi.sourceforge.net/).

5. Results

5.1. Inquiry 1: Number of Abbreviation and Synonym Definitions

SNOMED CT contains 357 095 named classes of which 131 954 (about 37%) qualify as abbreviations according to Definition 1. For 128 482 of these classes, SNOMED CT contains an *explicit* simple definition according to Definition 3. Otherwise, there are no explicit definitions of either abbreviations or synonyms. For 129 758 classes, only simple definitions can be inferred and for the remaining 2 196 classes, ambiguous definitions can be inferred. For 2 109 of these classes with implicit ambiguous definitions, SNOMED CT contains an explicit simple definition. This means that these 2 109 classes are equivalent to at least two complex class expressions, even though only one equivalence is stated explicitly. It is not clear whether this is intended in the design of SNOMED CT or not.

To summarise, we observe that the vast majority of abbreviations in SNOMED CT, i.e., more than 97%, are explicitly stated as simple definitions. There are only 3 472 classes for which no explicit definition can be found in SNOMED CT.

5.2. Inquiry 2: Use of Abbreviations and Synonyms

Table 1 shows how many times abbreviations are used in SNOMED CT. Each column reports on how many abbreviations are used a given number of times. For example, the last column tells us that there are only 10 explicitly defined abbreviations that are used over 100 times in SNOMED CT. It transpires that the vast majority of explicitly defined abbreviations are not used as such in SNOMED CT (cf. second left-most column in Table 1). Put differently, many complex class expressions are defined to be equivalent to a named class, but these classes do not occur in any other axioms in SNOMED CT.

Table 3
Recurrence of complex class expressions SNOMED CT.

Recurrence	1	2–4	5–10	11–50	51–100	101–1 000	1 001–5 000	> 5 000
Expressions	358 999	60 320	14 047	6 184	738	695	46	9

Yet, there are many abbreviations that are used at least once and up to ten times, namely $4\,994 + 3\,005 + 986 = 8\,985$. Furthermore, there is a fair number of abbreviations that are used more than ten times and up to 100 times, namely $285 + 103 + 23 = 411$. These numbers show that many named classes in SNOMED CT are used as abbreviations (according to Definition 4). Otherwise, we observe that the use of implicit abbreviations is comparable to the use of explicit abbreviations.

Table 2 shows how many times a complex class expression in SNOMED CT gives rise to a possible abbreviation use according to Definition 5. For example, the last column tells us that there are two explicitly defined abbreviations that could be reused more than 100 times in SNOMED CT. Here, we find that most complex class expressions associated with an explicit abbreviation do not give rise to any possible abbreviation use. This means that such complex class expressions do not occur SNOMED CT other than in an abbreviation's definition (which is not counted as a possible abbreviation use). However, there is also a fair number of complex class expressions that give rise to more than 10 possible abbreviation uses, namely $34+18+2+2 = 56$ in the case of explicitly defined abbreviations, and $42 + 4 + 1 + 1 = 48$ in the case of implicitly defined abbreviations.

5.3. Inquiry 3: Reoccurrence of Complex Expressions

Table 3 shows how often complex class expressions reoccur in the SNOMED CT. For example, the last column tells us that there are only nine complex class expressions in SNOMED CT that occur more than 5 000 times. Here, we find that there are 750 expressions that each occur more than a hundred times in SNOMED CT, some of which even occur more than a thousand times. Yet, there are only 24 abbreviations (10 explicit and 14 implicit) with more than a hundred uses (cf. Table 1). Furthermore, there are only three complex class expressions giving rise to more than a hundred possible abbreviation uses (cf. Table 2). This means that there are many complex class expressions without a defined abbreviation in SNOMED CT that reoccur often. Such complex class expressions can be considered as abbreviation candidates (cf. Definition 6). We will discuss the challenges of introducing abbreviations for complex class expressions in Section 7.

Comparing the total number of complex class expressions in SNOMED CT, 441 038, to the number of abbreviations, 131 954 (cf. Section 5.1), reveals that about 30% of complex class expressions are associated with an abbreviation.

5.4. Inquiry 4: Size and Nesting Depth of Complex Class Expressions

Figure 3 shows both the size and depth of complex class expressions occurring more than 50 times in SNOMED CT. It transpires that such complex class expressions are not deeply nested. In fact, there are only 39 cases with a nesting depth of three and 232 with a nesting depth of

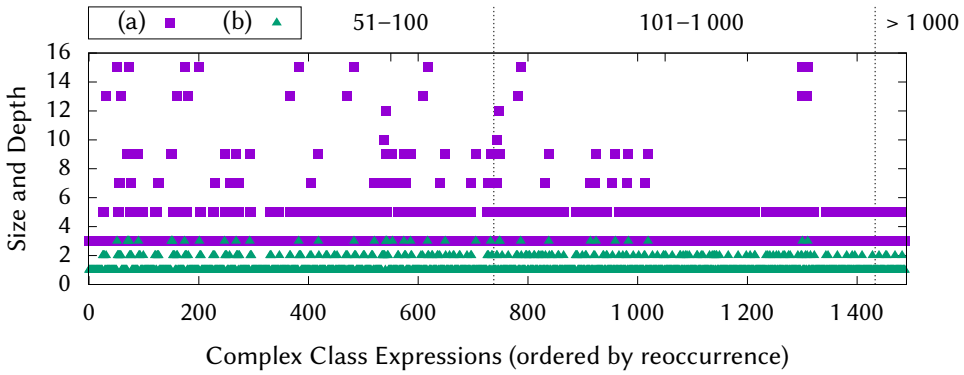


Figure 3: Size (a) and Depth (b) of Complex Class Expressions reoccuring 51–100, 101–1 000, and more than 1 000 times. Each Complex Class Expressions is given an index (shown on the x-axis) based on an ordering of their reoccurrence in SNOMED CT.

two, while the remaining 1 217 exhibit a nesting depth of 1. Similarly, the size of most such complex class expressions is comparatively small: 1 217 expressions exhibit a size of three, 195 exhibit a size of 5, and only 76 expressions have a size larger than 5.

However, it is important to note that there are instances of large complex class expressions with many occurrences in SNOMED CT. For example, the complex class expression at the index 1 311 in Figure 3 has a size of 15 and occurs 395 times. Expressions of this size are arguably hard to read and comprehend and might be good candidates for the introduction of an abbreviation.

6. Related Work

Logical equivalent rewritings for ontologies are usually motivated for the purpose of improved reasoning performance [12] or ontology-based data access [13]. However, the idea of rewriting axioms to improve ontology comprehension has also been discussed. Existing work in this direction focuses on rewritings that are minimal in size because large expressions are arguably hard to read and comprehend [14, 15, 2]. Yet, it is debatable whether the smallest possible logical rewriting of an axiom is indeed most suitable for human interpretation.

In the work presented in this paper, the focus is not on rewritings that are minimal in size. Rather, we study to what extent domain-specific vocabulary defined in an ontology can be *reused* to simplify otherwise complex expressions. The main argument being that a meaningful name is more readily understood by domain experts compared to its technical representation in a knowledge representation language, i.e., a complex class expression in OWL. It is important to note that the associated reduction in size is secondary in this context.

The task of determining abbreviations in an ontology (cf. Section 3.3) can be interpreted as concept definability, i.e., the problem of finding a definition for a concept name in an ontology [16]. However, in the context of this work, the problem of finding abbreviations is restricted to finding definitions for concepts in terms of complex class expressions that already occur, syntactically speaking, in an ontology. Nevertheless, advances in research on concept

definability may provide useful insights, e.g., knowing under what conditions implicitly defined concepts can also be defined explicitly. Furthermore, techniques for finding and comparing concept definitions are relevant in the context of suggesting abbreviations for suitable candidates (cf. Section 6).

7. Discussion & Outlook

So far, research into logical rewritings of ontologies for improving their comprehension has focused on reducing the size of logical expressions. However, this strategy may not always be appropriate in practice. For example, an axiom of the form $EquivalentClasses(\mathsf{A}, \mathsf{C}_1, \mathsf{C}_2)$ for a named class A may in fact capture two different ways of interpreting A, namely C_1 and C_2 — both of which may be useful in different contexts. In such a case, replacing all occurrences of both C_1 and C_2 in an ontology with A (excluding its definition), would lead to a more concise representation but doing so would remove information about the two different ways of interpreting A in different contexts.

So, the question of *when* to use an abbreviation may be subject to pragmatic considerations that may vary between use-cases. Developing an understanding of what makes a logical expression easier to understand or use in different contexts will require an evaluation of different rewriting strategies with user studies. Some work in this direction already exists for understanding justifications of entailments [17].

Besides the question of *when to use* an abbreviation, is the question of *when to introduce* a new abbreviation. We have proposed the notion of an abbreviation candidate (cf. Definition 6) that is based on the number of occurrences of a complex expression in an ontology. However, heeding our caution against using conciseness as the primary goal for rewriting ontologies, we also caution against introducing abbreviations solely based on their potential to replace many occurrences of a complex expression. Rather, many occurrences of a complex expression should only be seen as an indication for an abbreviation candidate.

Orthogonal to the introduction of a new abbreviation is the choice of its name. This name should be concise but informative so that it conveys relevant parts of its formal representation. The problem of finding a suitable name for a complex class expression involves two separate tasks. First, the complex class expression needs to be *verbalised* into a description of what the expression represents. And second, given such a description, one needs to find a word that matches the description. Intuitively, the second task can be seen as the inverse operation of a dictionary lookup. Instead of looking up the definition of a given word, one looks for a word given a description. As there already exists work for both of these two tasks, i.e., verbalising OWL expressions [18], on the one hand, and finding words given a query description [19], on the other hand, future research into combining both tasks for the purpose of suggesting suitable abbreviation names for complex OWL expressions seems promising.

Lastly, it needs to be highlighted that the introduction of an abbreviation, as defined in this work, *changes the meaning* of an ontology. Consider the ontology $\mathcal{O}$ and $\mathcal{O}_A = \mathcal{O} \cup \{\alpha\}$ where $\alpha = EquivalentClasses(\mathsf{A}, \mathsf{C})$ is a definition for an abbreviation A. If A does not occur in $\mathcal{O}$, then $\mathcal{O} \not\equiv \mathcal{O}_A$ because $\mathcal{O}_A \models \alpha$ but $\mathcal{O} \not\models \alpha$. This change in meaning can be avoided by encoding abbreviations using a meta-language, e.g., OTTR [20], on top of OWL. As an example,

consider the ontology

$$\mathcal{O} = \{ \begin{array}{ll} \text{Napoletana} & \text{SubClassOf Pizza and hasCountryOfOrigin value Italy}, \\ \text{Diavola} & \text{SubClassOf Pizza and hasCountryOfOrigin value Italy}, \\ \text{Hawaiian} & \text{SubClassOf Pizza and hasCountryOfOrigin value Canada} \end{array} \}$$

With OTTR, a mapping $\texttt{ItalianPizza} \mapsto \text{Pizza and hasCountryOfOrigin value Italy}$ can be defined, so that $\mathcal{O}$ can be encoded as

$$\mathcal{O}_T = \{ \begin{array}{ll} \text{Napoletana} & \text{SubClassOf } \texttt{ItalianPizza}, \\ \text{Diavola} & \text{SubClassOf } \texttt{ItalianPizza}, \\ \text{Hawaiian} & \text{SubClassOf Pizza and hasCountryOfOrigin value Canada} \end{array} \}$$

Note that `ItalianPizza` is not an OWL class but an expression in OTTR. In particular, the ontology $\mathcal{O}$ is semantically equivalent to $\mathcal{O}_T$ because the OTTR expression `ItalianPizza` is *indistinguishable* from Pizza and hasCountryOfOrigin value Italy on the level of OWL. The use of a meta-level language also opens up possibilities to summarise representations on higher levels of abstractions. In the case of the example ontology $\mathcal{O}$, the representation of a pizza's country of origin could be captured by a *parametrised* OTTR expression $\texttt{PizzaWithOrigin}(x) \mapsto \text{Pizza and hasCountryOfOrigin value } x$. With this, all three pizzas in $\mathcal{O}$ can be encoded in a uniform manner giving rise to the following even more succinct encoding:

$$\mathcal{O}_P = \{ \begin{array}{ll} \text{Napoletana} & \text{SubClassOf } \texttt{PizzaWithOrigin}(\text{Italy}), \\ \text{Diavola} & \text{SubClassOf } \texttt{PizzaWithOrigin}(\text{Italy}), \\ \text{Hawaiian} & \text{SubClassOf } \texttt{PizzaWithOrigin}(\text{Canada}) \end{array} \}$$

8. Conclusion

In this paper, we proposed a method for analysing how names in ontologies are defined and used. We used this method to conduct a use case study on SNOMED CT. We find that a large proportion, namely 37%, of named classes are defined to be equivalent to more complex class expressions (cf. Section 5.1). This finding suggests that concrete names for more complex ideas play an important role in SNOMED CT's design.

Furthermore, we find that such names are often reused (cf. Section 5.2). This finding suggests that concrete names for complex ideas seem to be preferred in SNOMED's design compared to their more complex logical representations. However, we also found a number of instances in which this is not the case. Whether this is intended or whether these cases are undesired effects of an automated workflow in which complex class expressions are generated will need to be determined by further research.

Lastly, we investigate the phenomenon of reoccurring complex class expressions that are not associated with a dedicated name. Here, we find that there are cases of complex class expressions that occur frequently in SNOMED CT and are non-trivial in both size and depth (cf. Section 5.3 and Section 5.4). These findings motivate further research into generating concise but informative names for complex class expressions for the purpose of improving ontology comprehension.

References

[1] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, U. Sattler, OWL 2: The next step for OWL, Journal of Web Semantics 6 (2008) 309–322.

[2] N. Nikitina, P. Koopmann, Small is beautiful: Computing minimal equivalent EL concepts, in: AAAI, AAAI Press, 2017, pp. 1206–1212.

[3] SNOMED International, https://www.snomed.org/, 2022. Accessed: July 11, 2022.

[4] B. Motik, P. Patel-Schneider, B. Parsia, OWL 2 Web Ontology Language. Structural Specification and Functional-Style Syntax (Second Edition) (2012). URL: http://www.w3.org/TR/owl2-syntax/.

[5] M. Horridge, P. Patel-Schneider, OWL 2 Web Ontology Language Manchester Syntax (Second Edition) (2012). URL: https://www.w3.org/TR/owl2-manchester-syntax/.

[6] abbreviation, n., in: OED Online, Oxford University Press, 2022. URL: https://www.oed.com/view/Entry/180?redirectedFrom=abbreviation&, accessed July 05, 2022.

[7] synonym, n., in: OED Online, Oxford University Press, 2022. URL: https://www.oed.com/view/Entry/196522?result=1&rskey=EVWsim&, accessed July 05, 2022.

[8] S. Bail, B. Parsia, U. Sattler, Extracting finite sets of entailments from OWL ontologies, in: Description Logics, volume 745 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2011.

[9] F. Baader, B. Hollunder, B. Nebel, H. Profitlich, E. Franconi, An empirical analysis of optimization techniques for terminological representation systems, or making KRIS get a move on, in: KR, Morgan Kaufmann, 1992, pp. 270–281.

[10] B. Glimm, I. Horrocks, B. Motik, R. D. C. Shearer, G. Stoilos, A novel approach to ontology classification, J. Web Semant. 14 (2012) 84–101.

[11] Y. Kazakov, M. Krötzsch, F. Simancik, The incredible ELK - from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies, J. Autom. Reason. 53 (2014) 1–61.

[12] D. Carral, C. Feier, B. C. Grau, P. Hitzler, I. Horrocks, *EL*-ifying ontologies, in: IJCAR, volume 8562 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 464–479.

[13] M. Imprialou, G. Stoilos, B. C. Grau, Benchmarking ontology-based query rewriting systems, in: AAAI, AAAI Press, 2012.

[14] F. Baader, R. Küsters, R. Molitor, Rewriting concepts using terminologies, in: KR, Morgan Kaufmann, 2000, pp. 297–308.

[15] M. Horridge, B. Parsia, U. Sattler, Laconic and precise justifications in OWL, in: ISWC, volume 5318 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 323–338.

[16] B. ten Cate, E. Franconi, I. Seylan, Beth definability in expressive description logics, J. Artif. Intell. Res. 48 (2013) 347–414.

[17] M. Horridge, S. Bail, B. Parsia, U. Sattler, The cognitive complexity of OWL justifications, in: Description Logics, volume 745 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2011.

[18] K. Kaljurand, N. E. Fuchs, Verbalizing OWL in attempto controlled english, in: OWLED, volume 258 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2007.

[19] F. Qi, L. Zhang, Y. Yang, Z. Liu, M. Sun, Wantwords: An open-source online reverse dictionary system, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 175–181.

[20] M. G. Skjæveland, D. P. Lupp, L. H. Karlsen, H. Forssell, Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates, in: ISWC (1), volume 11136 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 477–494.