Data in brief 29 (2020) 105098

Contents lists available at ScienceDirect

Data in brief

journal homepage: www.elsevier.com/locate/dib

Data Article

Dataset of ontology competency questions to SPARQL-OWL queries translations

Jedrzej Potoniec [a, b, *], Dawid Wiśniewski [a], Agnieszka Ławrynowicz [a, b], C. Maria Keet [c]

[a] Faculty of Computing, Poznan University of Technology, Ul. Piotrowo 3, 60-965 Poznan, Poland
[b] Center for Artificial Intelligence and Machine Learning, Poznan University of Technology, Ul. Piotrowo 2, 60-965 Poznan, Poland
[c] Department of Computer Science, University of Cape Town, Private Bag X3 Rondebosch 7701 South Africa

ARTICLE INFO

Article history:
Received 21 November 2019
Received in revised form 20 December 2019
Accepted 26 December 2019
Available online 7 January 2020

Keywords:
Ontology authoring
Competency questions
SPARQL-OWL
Semantic web

ABSTRACT

This data article reports on a new set of 234 competency questions for ontology development and their formalisation into a set of 131 SPARQL-OWL queries. This is the largest set of competency questions with their linked queries to date, covering several ontologies of different type in different subject domains developed by different groups of question authors and ontology developers. The dataset is focused specifically on the ontology TBox (terminological part). The dataset may serve as a manually created gold standard for testing and benchmarking, research into competency questions and querying ontologies, and tool development. The data is available in Mendeley Data. Its analysis is presented in "Analysis of Ontology Competency Questions and their formalizations in SPARQL-OWL" [15].

DOI of original article: https://doi.org/10.1016/j.websem.2019.100534.
* Corresponding author. Faculty of Computing, Poznan University of Technology, ul. Piotrowo 3, 60-965 Poznan, Poland.
E-mail addresses: Jedrzej.Potoniec@cs.put.poznan.pl (J. Potoniec), dawid.wisniewski@cs.put.poznan.pl (D. Wiśniewski), agnieszka.lawrynowicz@cs.put.poznan.pl (A. Ławrynowicz), mkeet@cs.uct.ac.za (C.M. Keet).

https://doi.org/10.1016/j.dib.2019.105098

Specifications Table

Subject	Computer science (Artificial Intelligence)
Specific subject area	Semantic Web
Type of data	Text files
How data were acquired	Systematic, manual translation from competency questions formulated in natural language to SPARQL-OWL
Data format	RQ files containing SPARQL-OWL queries
Parameters for data collection	Total count of CQs (234), total count of SPARQL-OWL queries (131), count of translatable CQs (131), count of untranslatable CQs (103)
Description of data collection	We selected only such ontologies that have competency questions stated against ontology schema (T-Box), deduplicated the competency questions where applicable, added contextual information, cleaned unnecessary additional notes and added markers for dematerialised (i.e., having variables in the sentence) versus materialised (i.e., with no variables) competency questions.
Data source location	Faculty of Computing Poznan University of Technology ul. Piotrowo 3 60-965 Poznan, Poland Department of Computer Science University of Cape Town Private Bag X3 Rondebosch 7701 South Africa
Data accessibility	Repository name: Mendeley Data Data identification number: https://doi.org/10.17632/pp6hmfxgfg.1 Direct URL to data: https://data.mendeley.com/datasets/pp6hmfxgfg/1
Related research article	Dawid Wiśniewski, Jedrzej Potoniec, Agnieszka Ławrynowicz, C. Maria Keet, Analysis of Ontology Competency Questions and their formalizations in SPARQL-OWL, Journal of Web Semantics, https://doi.org/10.1016/j.websem.2019.100534

Value of the Data

- The presented dataset of competency questions and their corresponding SPARQL-OWL queries for their respective ontologies is useful, because it constitutes the first gold-standard pipeline for ontology requirements verification.
- The provided dataset is also useful, because it enables systematic analysis what is translatable, what is not, and what are the reasons for why some of the questions were not translatable. This, in turn, enables automatic verification of competency questions provided by a user.
- The dataset may be beneficial for people concerned with research and development of methodologies and tools for ontology development and knowledge-based question answering tools.
- The dataset can be used for further research on, among others, competency question specification languages, query language optimization, automation of ontology authoring, and new ontology development methodologies.
- The dataset can provide further insights into the linguistic structure of competency questions and types of queries and how to formalise the former by means of the latter, which may facilitate tool development that will support the use of competency questions in the ontology development process.
- An additional value of the dataset is that it may be used to generate a benchmark, such as for automated competency question formalisation into SPARQL-OWL queries and for natural language generation of questions. The data set may also be used for a benchmark for a new knowledge-based question answering problem: question answering over ontology schema, since we provide translations from natural language competency questions to formal, and structured SPARQL-OWL queries.

1. Data description

Competency questions (CQs) [1] specify the knowledge that has to be entailed in the ontology and thus can be considered to be *requirements* on the content of the ontology. Although a CQ specification step is included in a few ontology engineering methodologies [2,3], CQs are hardly published. The notable comprehensive CQ sets for specific ontologies are for the Software Ontology [4] and Dem@Care

[5], which the majority of instances – if CQs are published at all – a couple of illustrative CQs in ontology papers (e.g. Ref. [6,7]). This hampers the development of methods and tooling support for CQs, which may hamper specification of CQs, and there with proper use of methodologies.

At the time of writing, there is no dataset of CQs from multiple ontologies available to take advantage of for CQ research and conclusions are being drawn from 1 to 2 CQ sets [8] and observation [9]. The claimed 248 CQs mentioned in Ref. [10] has 241 CQs on its website,[1] which are mostly for the *Vicinity core model ontology*, i.e., they cover one domain only. Using CQs for one ontology (or network of ontologies) may face the issue of bias, both due to the CQ authors thinking in one direction and, due to the subject domain, that may lend itself better for one or another type of CQ.

Second, the about 15–20 CQ patterns currently available are of the type *Which [CE1] [OPE] [CE2]?* [8] and similar [9], which mixes natural language with ways of modeling knowledge in the ontology. Also, there is no clear formalisation into SPARQL or SPARQL-OWL queries, nor a list of CQs and corresponding queries, not even for CQChecker (9) that translates the CQ into a Description Logic query if it is a TBox query, and executes a SPARQL query if it is an ABox query. Thus, even if one has the CQs, there is no accompanying set of queries to make them operationable. Likewise, the link of how to systematically go from the CQs to the 10 *test expression types* of [2] is missing.

In sum, there is no cross-domain, multi-author/group, dataset of CQs with broad coverage, systematics of how to formalise the CQs – be this manually or automatically – is lacking, and a corresponding set of queries over the ontologies is missing. Yet, they are essential ingredients for research into CQs.

To alleviate this we selected 5 ontologies with 234 competency questions in total: 88 for Software Ontology (SWO) [4], 11 for Stuff ontology (Stuff) [11], 14 for African Wildlife Ontology (AWO) [12], 107 for Dementia Ambient Care ontology (Dem@Care) [5], 14 for Ontology of Datatypes (OntoDT) [13]. We manually translated these CQs to SPARQL-OWL [14] and we share a set of 131 SPARQL-OWL queries obtained this way. Its analysis is presented in "Analysis of Ontology Competency Questions and their formalizations in SPARQL-OWL" [15].

In the remainder of the paper, we present and discuss multiple SPARQL-OWL queries. To keep them easily readable, we use prefixes to replace full namespaces in URIs, but for brevity, we omit the preambles of the queries. Instead, we provide the full list of the prefixes used and their corresponding namespaces in Table 1.

Table 1
Prefixes used in the dataset description.

prefix	namespace
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
owl:	http://www.w3.org/2002/07/owl#
xsd:	http://www.w3.org/2001/XMLSchema#
awo:	http://www.meteck.org/teaching/ontologies/AfricanWildlifeOntology1.owl#
stuff:	http://www.meteck.org/files/ontologies/stuff.owl#
event:	http://www.demcare.eu/ontologies/event.owl#
exch:	http://www.demcare.eu/ontologies/exchangemodel.owl#
home:	http://www.demcare.eu/ontologies/home.owl#
lab:	http://www.demcare.eu/ontologies/lab.owl#
swo:	http://www.ebi.ac.uk/swo/
efo-swo:	http://www.ebi.ac.uk/efo/swo/
maturity:	http://www.ebi.ac.uk/swo/maturity/
interface:	http://www.ebi.ac.uk/swo/interface/
obo:	http://purl.obolibrary.org/obo/
OntoDT:	http://www.ontodm.com/OntoDT#
OntoDT2	http://ontodm.com/OntoDT#

[1] http://vicinity.iot.linkeddata.es/vicinity/testing.html.

2. Experimental design, materials, and methods

2.1. Competency questions preparation

The dataset of CQs was prepared for the translation to SPARQL-OWL queries in the following way:

1. For each CQ set defined against same ontology – if duplicates consisting of exactly the same words were identified – they were all reduced to single instance
2. For every question where important context was missing (for example: „What is the input and output?" does not define which entity input and output should be considered) – the most probable context was added. Thus, the question: „What is the input and output" became „What is the input and output for this software?"
3. If unnecessary additional notes were detected, like content of the brackets in the following CQ: *(+ version) What data from person x is analysed with tool y, version z?* – the additional notes were removed
4. If the question contained a fragment of text, that is too general to provide an answer but at the same time the ontology defines subtypes or instances of that fragment, we marked such fragments with square brackets. This was then used during the translation to SPARQL-OWL to recognize parts that cannot be represented with a URI in the query and must be represented with a placeholder instead. We call such CQ dematerialised, as they cannot be answered directly, but they can be used to generate multiple materialised questions. For example CQ *What is the algorithm used to process [this data]?* can only be answered once *this data* is specified, e.g., leading to questions such as: *What is the algorithm used to process CSV data?* or *What is the algorithm used to process XML data?.*

2.2. Translating competency questions to SPARQL-OWL

To obtain the formal representation of gathered CQs we manually translated them to SPARQL-OWL. The process of translating a CQ was organized as follows. First, we identified keywords in the CQ, which then were used to identify relevant vocabulary in the ontology. If we could not find a match for a given keyword, we looked for different surface forms and words with similar meaning. Next, we identified a subset of expected answers for the query, either by identifying relevant vocabulary using the answers provided with the CQ, or by trying to answer the CQ using the knowledge present in the ontology. Then, we used the identified vocabularies to decide how to construct the query in such a way, that it would provide the expected answers. This step required careful inspection of the modeling style of the ontology. Finally, the query was constructed and tested using OWL-BGP,[2] to verify whether it was constructed correctly and yields the expected answers.

During the process we observed a high variability in the structure of the resulting queries and we identified two main causes: (i) the CQs themselves can vary significantly within and between ontologies; (ii) knowledge in the ontology can be modeled in various ways, sometimes highly diverging from the "default" approaches suggested by the OWL standards. Below, we discuss the obtained queries in detail, providing justification for the decisions made and highlighting the most interesting cases.

Some of the questions are not context-independent, in this sense that they contain unresolvable pronouns (e.g., *this software* in *What is the valid input for this software?*) or placeholders (e.g., *this stuff* in *Is [this stuff] a pure or a mixed stuff?*). We performed the translation introducing placeholders in the queries to represent the placeholders/pronouns from the question and we denote them in the graph patterns using variables surrounded with $ instead of prefixed with ?.

[2] https://github.com/iliannakollia/owl-bgp.

2.3. African Wildlife Ontology

For African Wildlife Ontology (AWO) we gathered 14 competency questions. Six of them (awo_5, awo_9—awo_13), concerning drinking, habitats and conservation status, were deemed impossible to translate due to the lack of vocabulary in the ontology. Four questions (awo_1, awo_6—awo_8) represented the same pattern of asking about classes connected with a property specified in the questions using existential restriction. For example, awo_6 *Which plants eat animals?* was translated as

```
SELECT DISTINCT ?eats
WHERE {
?eats rdfs:subClassOf awo:plant, [
     a owl:Restriction ;
     owl:onProperty awo:eats;
     owl:someValuesFrom awo:animal
] .
FILTER(?eats != owl:Nothing)
}
```

The query asks for all subclasses of the class `awo:plant`, which has existential restriction on the property `awo:eats` targeting the class `awo:animal`. Because SPARQL-OWL uses reasoning during answering the query, the answer for the query will contain classes with more specific restrictions (e.g., `awo:impala`, which is a subclass of `awo:animal`). The filter clauses was added to remove, technically correct, but useless, answer of `owl:Nothing`, i.e., the bottom concept.

Awo_2 *Is [this animal] a herbivore?* is a simple question to check whether a class is a subclass of another class, while awo_4 *Does a lion eat plants or plant parts?* is similar, but yields a more complicated query: the superclass in the query is an existential restriction on the property `awo:eats` and plants or plant parts must be modeled as a union of classes `awo:plant` and `awo:PlantParts`. Awo_14 Are there animals that are carnivore but still eat some plants or parts of plants? can be interpreted in two ways. One possibility is to treat the question as a question about presence of particular knowledge in the ontology or, in other words, Does the ontology contain any carnivore that eats some plants or parts of plants?. In this interpretation the question resembles the formerly discussed questions awo_1 and awo_6—awo_8. The other possibility is to interpret the question as a question about possibility of existence of such an animal or, in other words, Is it possible for a carnivore to eat some plants or parts of plants? Then the questions corresponds to the query about (lack of) a disjointness axiom between the class `:carnivore` and the existential restriction already discussed in awo_4. Because the presence of the disjointness axiom corresponds to the negative answer to the original question, the query must contain negation, which can be obtained using `FILTER NOT EXISTS`.

2.4. Stuff

For the Stuff ontology, we collected 11 competency questions that can be divided into 6 distinct categories with respect to the required reasoning. Stuff_01 *Is [this stuff] a pure or a mixed stuff?* is a question to decide whether a class (e.g., Mayonnaise) is a subclass of one of the two specified classes: *PureStuff, MixedStuff*. Stuff_02 What *is the difference between [this colloid] and [this colloid]?* and stuff_08 are questions about finding differences between definitions of the classes. In principle, the ontology contains enough information to answer the questions, as the referred classes have their complete definitions there. Unfortunately, this is a non-standard reasoning task and is not possible to express it using SPARQL-OWL. Stuff_03 *In which phases are the stuffs in [this colloid]?* requires finding all the superclasses of a specified class (e.g., emulsion), that are complex class expression with nested existential restrictions on properties, respectively, `stuff:hasPartStuff` and `stuff:hasState`. The class expressions from the nested restriction are the answers to the query. Stuff_04 Can a solution be a pure stuff? and stuff_06 asks whether two classes are disjoint. Stuff_04 requires negation, similarly to awo_04 from AWO. Stuff_05 Which kind of stuff are [these stuffs]?, stuff_09 and stuff_10 are questions

about all the superclasses of a given class (Stuff_05: these stuffs, e.g., emulsion), that are subclasses of another class (Stuff_05: `stuff:Stuff`). Stuff_11 Where do I categorise bulk like [this bulk]? is a simpler version, that does not contain the second requirement Stuff_07 Which stuffs have as part exactly two substuffs? is a question about all the subclasses of a class expression using cardinality restriction $= 2$ on the property `stuff:hasSubStuff`.

2.5. Dem@Care

The Dem@Care ontology is accompanied by 107 competency questions and (a subset of) expected answer for each of them. 47 of them lack the appropriate vocabulary and/or knowledge in the ontology and/or are too vague and thus cannot be modeled. The remainder can be divided into six groups depending on the shape of the query.

Group I contains three questions (DemCare CQs: 4, 6, 8) asking for instances of a given class, e.g., demcare_4 What *is the gender information?* corresponds to the BGP:

```
?x rdf:type lab:GenderType.
```

Group II contains 17 questions (DemCare CQs: 7, 23, 29–32, 57, 65, 72, 75, 77, 78, 82, 87, 105–107) that ask for all the proper subclasses of a given class, e.g., the BGP of demcare_7 What *types of clinical data are collected?* is

```
?x rdfs:subClassOf lab:ClinicalAssessment .
FILTER(?x != lab:ClinicalAssessment)
```

where `?x` is the distinguished variable and the filter clause is to ensure only proper subclasses are considered.

Group III, containing 7 questions (DemCare CQs: 83–85, 88, 90, 98, 104), corresponds to questions asking about the direct subclassess of a given class. For example, demcare_83 What *are the main types of entities?* corresponds to the following graph pattern:

```
?x rdfs:subClassOfevent:Entity .
FILTER NOT EXISTS {
     ?x rdfs:subClassOf ?y .
     ?y rdfs:subClassOfevent:Entity.
     FILTER(?y != event:Entity&& ?x != ?y)
}
FILTER(?x != event:Entity&& ?x != owl:Nothing)
```

We distinguished Group II from Group III by, respectively, absence or presence the word „main" (cf., the main types above) in the question.

Group IV is by far the largest, containing 24 questions (DemCare CQs: 3, 9, 15, 19, 21, 40, 47, 50, 52–54, 58–60, 62–64, 68, 76, 80, 89, 99, 101, 103). These are questions about relations an object of a given class is expected to have, i.e., about property names present in existential restrictions. For example, demcare_3 What *types of demographic data are collected?* can be represented by the following graph pattern:

```
lab:DemographicCharacteristicsRecord rdfs:subClassOf [
     a owl:Restriction ;
     owl:onProperty ?x;
     owl:someValuesFrom []
].
```

where `lab:DemographicCharacteristicsRecord` is the class named in the query, and `?x` is the distinguished variable. There is also another possibility to represent questions from this category. Consider demcare_89 What are the main types of information describing an event? and its gold answer: The agent of the event (i.e. the referred person, object or room), start time, duration, and location (where applicable). The where applicable part hints that there are properties that are expected only for some subtypes of an event (i.e., subclasses of the class `event:Event`). We can thus consider the following graph pattern

```
[] rdfs:subClassOf event:Event, [
     owl:onProperty ?p;
     owl:someValuesFrom []
].
```

Here, we consider all the subclasses of the class `event:Event` and ask for properties they are expected to have. Queries of such form are more general than the queries of the earlier form in the sense that the latter always returns at least the same information as the earlier (query subsumption).

Group V contains 7 questions (DemCare CQs: 33–39) that ask about class names of values for a specified property. For example, demcare_33 What *is assessed in the walking task?* can be represented as a query with the following graph pattern:

```
lab:S1_P11_WalkingTaskrdfs:subClassOf [
     a owl:Restriction;
     owl:onPropertylab:measuredData;
     owl:someValuesFrom ?x
].
?x rdfs:subClassOf lab:MeasuredData.
FILTER(?x != lab:MeasuredData)
```

The first triple pattern selects all the class names that are present in an existential restriction for the class `lab:S1_-_P11_WalkingTask` (or its ancestors) on the property `lab:measuredData`, while the second along with the filter clause ensure that only proper subclasses of the class lab:MeasuredData are returned. Should the second part be omitted, the query would return also, e.g, `owl:Thing`, which certainly does not answer the question.

Group VI contains 2 questions that are classes on their own and require detailed discussion. demcare_67 What *information is of clinical interest regarding food and drink preparation?* is a union of two questions from the group IV (questions about property names) with two different classes: `event:PrepareMeal` and `event:PrepareDrink`. It can be realized using SPARQL UNION clause or using OWL `owl:unionOf`, and the graph pattern of the first possibility is presented below.

```
{
     [] rdfs:subClassOfevent:PrepareMeal, [
          owl:onProperty ?p;
          owl:someValuesFrom []
     ].
} UNION {
     [] rdfs:subClassOfevent:PrepareDrink, [
          owl:onProperty ?p;
          owl:someValuesFrom []
     ].
}
```

The second question in the group demcare_100 What *are the main types of data a report may refer to?*. Its expected answers are *Questionnaires, clinical characteristics, demographic data, ...* and they can indeed be found in the axioms describing the class `exch:Report`, namely in a universal restriction on the property `exch:describes`. To retrieve them all it is necessary to dig into the RDF list representation of the union, as presented in the BGP below.

```
exch:Reportrdfs:subClassOf [
     a owl:Restriction ;
     owl:onProperty exch:describes ;
     owl:allValuesFrom [
          a owl:Class ;
          owl:unionOf/rdf:rest*/rdf:first ?c
] ].
```

2.6. Software Ontology

The Software Ontology SWO is accompanied by the set of 88 competency questions. Out of these 88 questions, 46 were deemed impossible to translate to SPARQL-OWL, either due to the absence of relevant vocabulary in the ontology or due to their ambiguity. The remaining questions can be divided into 9 groups based on the shape of the graph pattern in the final query. The division is coarse and aimed at underlying recurring patterns rather than at introducing a strict classification.

Group I consists of questions about classes occurring in an existential restriction for a given class. For example, consider the question swo_04 *Which of the named and published "algo-rithms" does this tool use?*. This can be achieved with the following graph pattern:

```
$sw$ rdfs:subClassOf [
     owl:onProperty efo-swo:SWO_0000740;
     owl:someValuesFrom ?alg
] .
?alg rdfs:subClassOf obo:IAO_0000064 .
FILTER(?alg != obo:IAO_0000064 && ?alg != owl:Nothing).
```

`Swo:SWO_0000740` is an object property labeled `implements` and its range is the class `obo:IAO_0000064 algorithm`. The second subclass expression ensures that only algorithms are listed, otherwise also superclasses of the class `obo:IAO_0000064` would be returned. The filter expression is added to remove this particular class and the bottom concept `owl:Nothing` from the results, as they are both meaningless. CQs with similar form of the graph pattern are swo_07, swo_45, and swo_83.

Group II is very similar but requires retrieving an actual value (an individual or a literal) rather than a class name. Consider swo_36 What *is the homepage of the software?* which can be represented using the following graph pattern:

```
$sw$ rdfs:subClassOf [
     owl:onProperty swo:SWO_0004006 ;
     owl:hasValue ?url
]
```

The property `swo:SWO_0004006` is a data property `has website homepage`. The main difference between this graph pattern and the previous one is the usage of `owl:hasValue` instead of `owl:someValuesFrom` and the lack of the filter expression, which is not needed in this situation. The other questions following the same pattern are: swo_29 (approximation), swo_31, swo_39, swo_54 (uses an object property instead of a data property), swo_70, swo_72, and swo_73. An interesting corner case in this group is swo_76 Is there a publication with it?. The ontology itself does not provide appropriate vocabulary and in general this is impossible to answer. However, the property `swo:SWO_0000043 has documentation` is hacked in an interesting way: most of its usages in the ontology has values being URLs starting with `https://doi.org/`, thus referring to a publication with a DOI. An approximate answer to this particular question could be thus obtained using the following graph pattern:

```
$sw$ rdfs:subClassOf [
     owl:onProperty swo:SWO_0000043 ;
     owl:hasValue ?doc
].
FILTER(STRSTARTS(?doc, "http://dx.doi.org/"))
```

To answer literally for the posed query the pattern should be used in an ASK query, but it can be as well used in a SELECT query to produce a list of publications instead. Another interesting variant is the question swo_44 *How long has this software been around?* which requires postprocessing of the answer. Consider the following graph pattern:

```
$sw$ rdfs:subClassOf [
     owl:onProperty maturity:SWO_9000068 ;
     owl:hasValue ?date
]
BIND(now()-xsd:dateTime(?date) AS ?result)
```

This is a pattern typical for the questions in this group plus a `BIND` expression to compute the difference between the release date and the current time, which is the expected result of the query.

Group III is constituted by questions of type *Which software ... ?*, e.g., swo_08 *Which software can perform task x?*. This is also similar to Group I, but the placeholder is placed in the superclass expression and the distinguished variable is the subclass position. The sample question can be realized with the following graph pattern:

```
?sw rdfs:subClassOf swo:SWO_0000001 , [
     owl:onProperty swo:SWO_0040005 ;
     owl:someValuesFrom $task$
] .
```

The class swo:SWO_0000001 software is in the query to ensure that only actual pieces of software are returned, while the existential restriction on the property swo:SWO_0040005 is executed in ensures that the additional condition of the question (i.e., performing task x) is fulfilled. The query can again be extended with filtering out the bottom concept. The other question in this group is swo_14.

Group IV consists of yes-no questions of type *Is this software ...*, e.g., swo_53 *Is this software available as a web service?*. This is a variant of Group III, because the only difference is in using a placeholder instead of the normal variable in the graph pattern and the appropriate SPARQL verb is ASK rather than SELECT. The sample question can be realized using the following graph pattern:

```
$sw$ rdfs:subClassOf [
     owl:onProperty swo:SWO_0004001 ;
     owl:someValuesFrom interface:SWO_9000051
] .
```

The property swo:SWO_0004001 is labeled has interface and the class swo:SWO_9000051 web service. The other questions in the group are: swo_9, swo_65. Also, swo_88 Do I need a license key to use it? can be approximated by this group. In general, the question is about a technical side of licensing and cannot be represented using the available vocabulary, but one can approximate it as a question about whether the license is proprietary or not.

Group V consists of conjunctive mixtures of the previous groups. For example, consider swo_11 *Which visualization software is there for this data and what will it cost?*. The ontology does not cover the area of software costs, but the remainder of the question, i.e., *Which visualization software is there for this data?* can be answered using a conjunction of two patterns from Group III. In other words, the question can be seen as a conjunction of two questions: *Which visualization software is there?* and *Which software is there for this data?*. This yields the following graph pattern:

```
?sw rdfs:subClassOf swo:SWO_0000001 ;
rdfs:subClassOf [
     owl:onProperty swo:SWO_0000086 ; # has specified data input
     owl:someValuesFrom ?data
] ;
rdfs:subClassOf [
     owl:onProperty swo:SWO_0040005 ; # is executed in
     owl:someValuesFrom efo-swo:SWO_0000724 # data visualization
] ;
FILTER(?sw != owl:Nothing)
```

The other similar questions are swo_15 and swo_57 (two patterns from Group II, one with a variable replaced by a placeholder).

Group VI are the questions that require a nested existential restriction, possibly in a conjunction with patterns from the previous groups. For example, consider swo_01 What *is the algorithm used to process this data?* The question is vague in this sense that, in principle, it is possible to use a sorting algorithm on an arbitrary binary data, but such an operation is usually meaningless and thus such an answer would be unexpected. Moreover, the ontology deals with the algorithms only to this extent that algorithms are implemented by something, but they are not described by themselves, in particular: their expected inputs and outputs are unknown. Finally, we assumed that this data refers to a data format. Under these assumptions the question could be rewritten as follows: *What are the algorithms implemented by software that is capable of processing data in this format?*, *this format* being a placeholder. This yielded quite a complex query presented below. The query looks for a piece of software (the class swo:SWO_0000001) capable of processing an input (the existential restriction on the property swo:SWO_0000086has specified data input) such that it is data (the class obo:IAO_0000027-data) and it is expressed in the given format (the existential restriction on the property

swo:SWO_0004002 has format specification). From this software, the implemented algorithms are extracted (the existential restriction on the property swo:SWO_0000740 implements). As we are interested only in the actual algorithms further filtering is performed by ensuring that the variable ?alg is bound to a proper subclass of the class obo:IAO_0000064 algorithm.

```
[] rdfs:subClassOf swo:SWO_0000001 ;
    rdfs:subClassOf [  owl:onProperty swo:SWO_0000086 ;
    owl:someValuesFrom [
         owl:intersectionOf (obo:IAO_0000027 [
              owl:onProperty swo:SWO_0004002 ;
              owl:someValuesFrom $format$
         ] )
    ]
] ;
    rdfs:subClassOf [
         owl:onProperty efo-swo:SWO_0000740;
         owl:someValuesFrom ?alg
] .
?alg rdfs:subClassOf obo:IAO_0000064 .
FILTER(?alg != obo:IAO_0000064) .
```

Similar questions are swo_18–21, swo_58, swo62, and swo_78. Group VII consists of the questions that require comparing (in a broad sense) two entities on a specified criterion. For example consider swo_02 What *are the alternatives to this software?*. Again, this question cannot be directly answered by the ontology, so we assumed that a software can be treated as an alternative for another software if there exists an algorithm implemented by both pieces of software. We obtained the following BGP:

```
$sw$ rdfs:subClassOf [
    owl:onProperty efo-swo:SWO_0000740;
    owl:someValuesFrom ?alg
] .
?alg rdfs:subClassOf obo:IAO_0000064 .
FILTER(?alg != obo:IAO_0000064) .
?alt rdfs:subClassOf swo:SWO_0000001, [
    owl:onProperty efo-swo:SWO_0000740;
    owl:someValuesFrom ?alg
] .
FILTER($sw$ != ?alt)
```

For the given software sw we extract the algorithms using the same approach as in the query for swo_01. We then look for pieces of software that implement any of these algorithms and ensure and that we provide an actual alternative, i.e., a different piece of software. Other questions in the group are swo_03 (a semantic duplicate of swo_02), swo_16, and swo_22.

Group VIII consists of the questions that required disjunction (i.e., owl:unionOf or SPARQL UNION) in their corresponding query. There were three such questions and we discuss reasons and usage of disjunction in them, but for sake of brevity, we abstain from providing them in the text. In the first of them, swo_25 What open source, maintained software can I use to process these in this format? the word maintained can actually be mapped to two terms from SWO: swo:SWO_9000065 Live and swo:SWO_9000073 Maintained, so we used owl:unionOf to include both. In swo_59 What license does it have and what is its permissiveness? we used union join two cases: the case with a license without any clauses describing it (i.e., without information about its permissiveness available), and with them. Finally, in swo_67 Is it free or not?, we use union to combine three separate license clauses

representing free software: `swo:SWO_9000030usage unrestricted`, `swo:SWO_9000020source code available` and `swo:SWO_1000059free`.

Group IX contains a single question swo_26 *Is the output format of it proprietary?*. The ontology does not deal with licenses of the data formats, so in principle the question cannot be answered. We provide quite a complex proxy based on an assumption that a data format is proprietary if there is no open source software capable of producing it. We approached it by using SPARQL FILTER NOT EXISTS clause with a BGP from Group VI inside. The interested reader is referred to the dataset for the full code of the query.

2.7. OntoDT

OntoDT is equipped with the set of 14 competency questions. One of them, ontodt_09, is an instance-level question and thus we did not translate it. Most of the remaining questions are concerned with classes related to each other by some property. For example, the question ontodt_01 What *is the set of characterizing operations for [a datatype X]?* was translated as

```
$X$ rdfs:subClassOf OntoDT2:OntoDT_487147, [
a owl:Restriction ;
owl:onProperty OntoDT:OntoDT_0000400 ;
owl:someValuesFrom ?x
]
```

where `$X$` corresponds to the placeholder [a datatype X] and `?x` is the result. The same pattern is exhibited by ontodt_02. Similar questions are: ontodt_03 and ontodt_10–ontodt_14, where we check whether `?x` is a subclass of a specified class, and ontodt_04–ontodt_05, where `$X$` and `?x` swapped places, i.e., we ask about subclasses of a particular class expression.

The question ontodt_06 What *is the set of datatypes that have [a datatype quality X] and [characterizing operation Y]?* also exhibits a similar pattern, but twice: we demand that the resulting classes are subclasses of two class expressions with an existential restriction. For question ontodt_07 What *are the aggregated datatypes that have [an aggregate generator property X]?* the resulting query is more complex, because we need to query for subclasses of an existential restriction with a nested existential restriction:

```
?x rdfs:subClassOf OntoDT2:OntoDT_378476, [
aowl:Restriction ;
owl:onProperty OntoDT:OntoDT_0000405;
owl:someValuesFrom [
aowl:Restriction ;
owl:onProperty obo:OBI_0000298;
owl:someValuesFrom $X$
]
] .
$X$ rdfs:subClassOf OntoDT2:OntoDT_283020 .
```

In this BGP `$X$` corresponds to the placeholder in the question [an aggregate generator property X], while `?x` is the distinguished variable. The query corresponding to the question ontodt_08 is very similar.

Acknowledgments

Jedrzej Potoniec acknowledges support from the grant 09/91/DSMK/0659. Dawid Wiśniewski and Agnieszka Ławrynowicz acknowledge support by the Polish National Science Center (Grant No 2014/13/D/ST6/02076).

Conflict of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Ontologies: principles, methods and applications in, M. Uschold, M.2 Gruninger, Knowledge Eng. Review, vol. 11, 1996, pp. 93–136.
[2] P.C. Barbosa Fernandes, R.S.S. Guizzardi, G. Guizzardi, Using goal modeling to capture competency questions in ontology-based systems, J. Inf. Data Manag. 2 (2011).
[3] Mari Carmen Suarez-Figueroa, Asuncion Gomez-Perez, Mariano Fernandez-Lopez, The NeOn Methodology framework: a scenario-based methodology for ontology development, Appl. Ontol. 10 (2015) 107–145.
[4] J. Malone, et al., The software ontology (SWO): a resource for reproducibility in biomedical data analysis, curation and digital preservation in, J. Biomed. Semant. 5 (2014).
[5] S. Dasiopoulou, G. Meditskos, V. Efstathiou, Semantic Knowledge Structures and Representation. s.L. : FP7-288199 Dem@Care: Dementia Ambient Care: Multi-Sensing Monitoring for Intelligence Remote Management and Decision Support, 2012.
[6] C. Maria Keet, et al., The data mining OPtimization ontology, J. Web Semant. 32 (May 2015) 43–53.
[7] Joao Moreira, et al., SAREF4health: IoT standard-based ontology-driven healthcare systems, in: Formal Ontology in Information Systems - Proceedings of the 10th International Conference, FOIS 2018, Cape Town, South Africa, 2018, 19-21 September 2018.
[8] Yuan Ren, et al., Towards competency question-driven ontology authoring, in: The Semantic Web: Trends, Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, 2014. May 25-29, 2014.
[9] Camila Bezerra, Filipe Santana, Fred Freitas, CQChecker: A Tool to Check Ontologies In OWL-DL Using Competency Questions Written In Controlled Natural Language, Learning & Nonlinear Models, 2014.
[10] Alba Fernandez-Izquierdo, Garcia-Castro Raul, Requirements Behaviour Analysis for Ontology Testing, in: Catherine Faron-Zucker, et al. (Eds.), Proceedings of the 21st International Conference on Knowledge Engineering, Knowledge Management (EKAW'18), Springer, 2018, pp. 114–130.
[11] C.M. Keet, A core ontology of macroscopic stuff, in: Krzysztof Janowicz, et al. (Eds.), Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW, Linkoping, Sweden, 2014, pp. 209–224. November 24-28, 2014. Proceedings.
[12] C.M. Keet, The African Wildlife Ontology Tutorial Ontologies: Requirements, Design, and Content, Technical Report 1905.09519, 23 May 2019. https://arxiv.org/abs/1905.09519.
[13] P. Panov, L.N. Soldatova, S. Dzeroski, Generic ontology of datatypes, Inf. Sci. 329 (2016) 900–920.
[14] I. Kollia, B. Glimm, I. Horrocks, SPARQL query answering over OWL ontologies, in: The Semantic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, 2011. May 29-June 2, 2011, Proceedings, Part I.
[15] D. Wiśniewski, J. Potoniec, A. Ławrynowicz, C. Maria Keet, Analysis of ontology competency questions and their formalizations in SPARQL-OWL, J. Web Semant. (2019), https://doi.org/10.1016/j.websem.2019.100534.