RDF Constraints Classification Ensuring High Quality of Metadata and Data

Thomas Bosch[1], Benjamin Zapilko[1], Joachim Wackerow[1], and Kai Eckert[2]

[1] GESIS – Leibniz Institute for the Social Sciences, Germany
`{firstname.lastname}@gesis.org`,
[2] University of Mannheim, Germany
`kai@informatik.uni-mannheim.de`

Abstract. For research institutes, data libraries, and data archives, RDF data validation according to predefined constraints is a much sought-after feature, particularly as this is taken for granted in the XML world. To ensure high quality and trust, metadata and data must satisfy certain criteria - specified in terms of RDF constraints.
In this paper, we propose a system to classify RDF constraints and RDF constraint types, which in most cases correspond to RDF validation requirements. Constraints are instantiated from constraint types in order to validate both metadata and data represented by any vocabulary. Within the context of a complex and complete real world running example within the community around research data for the *social, behavioural, and economic (SBE) sciences*, we prove the claim that the developed classification system perfectly applies for diverse vocabularies. We show how data in rectangular format (*PHDD*) and metadata on person-level data sets (*Disco*), aggregated data sets (*QB*), thesauri (*SKOS*), and statistical classifications (*XKOS*) are represented in RDF and how therefore reused vocabularies are interrelated. We explain how *SBE* (meta)data is validated against constraints to ensure high quality of and trust in (meta)data. We exhaustively evaluated the metadata quality of large real world aggregated (*QB*), person-level (*Disco*), and thesauri (*SKOS*) data sets (more than 4.2 billion triples and 15 thousand data sets) by means of constraints of the majority of the constraint types.

Keywords: RDF Validation, RDF Constraints, DDI-RDF Discovery Vocabulary, Disco, RDF Data Cube Vocabulary, Linked Data, Semantic Web

1 Introduction

For more than a decade, members of the community around research data for the *social, behavioural, and economic (SBE) sciences* have been developing and using a metadata standard (composed of almost twelve hundred metadata fields) known as the *Data Documentation Initiative (DDI)* [11]. *DDI* is an XML format designed to support the dissemination, management, and reuse of the data collected and archived for research purposes. Increasingly, data professionals, national statistical institutes, data archives, data libraries, and government statisticians (e.g. data.gov, data.gov.uk) are very interested in having their data be discovered and used by providing their metadata (e.g. about unemployment rates or income) on the web in form of RDF. Recently, members of the SBE and Linked Data community developed the *DDI-RDF Discovery Vocabulary (Disco)*[3], an effort to leverage the mature DDI metadata model for the purposes of exposing DDI metadata as resources within the Web of Linked Data.

For data archives, research institutes, and data libraries, RDF data validation according to predefined constraints is a much sought-after feature, particularly as this is taken for granted in the XML world as DDI-XML documents are validated against diverse XSDs[4]. Several approaches exist to meet this requirement, ranging from using *OWL 2* as a constraint language to *SPIN*[5], a SPARQL-based way to formulate and check constraints. There are also constraint languages like *Shape Expressions*, *Resource Shapes* or *Description Set Profiles* that more or less explicitly address the SBE community. In 2013, the W3C organized the RDF Validation Workshop[6], where experts from industry, government, and academia discussed first use cases for RDF constraint formulation and RDF data validation. In 2014, two working groups on RDF validation have been established to develop a language to express constraints on RDF data: the W3C RDF Data Shapes working group[7] and the DCMI RDF Application Profiles task group[8].

Bosch and Eckert [1] collected the findings of these working groups and initiated a database of RDF validation requirements which is available for contribution at http://purl.org/net/rdf-validation. The intention is to collaboratively collect case studies, use cases, requirements, and solutions regarding RDF validation in a comprehensive and structured way. The requirements are classified to better evaluate existing solutions and each requirement is directly mapped to a constraint type which may be expressed by at least one existing constraint language. Bosch and Eckert [2] use SPIN as basis to define a validation environment (available at http://purl.org/net/rdfval-demo) in which the validation of any constraint language[9] can be implemented by representing them in SPARQL. The SPIN engine checks for each resource if it satisfies all constraints, which are associated with its assigned classes, and generates a result RDF graph containing information about all constraint violations.

The main **contribution** of this paper is the development of a system to classify RDF constraints and RDF constraint types, which in most cases correspond to RDF validation requirements[10] (section 4). We propose an extensible metric

[3] http://rdf-vocabulary.ddialliance.org/discovery.html
[4] http://www.ddialliance.org/Specification/
[5] http://spinRDF.org/
[6] http://www.w3.org/2012/12/rdf-val/
[7] http://www.w3.org/2014/rds/charter
[8] http://wiki.dublincore.org/index.php/RDF-Application-Profiles
[9] The only limitation is that constraint languages must be represented in RDF
[10] For simplicity reasons, we use the terms *constraint types* and *constraints* instead of *RDF constraint types* and *RDF constraints* in the rest of the paper

to measure the continuum of severity levels to indicate how serious the violation of given constraints is. Constraints are instantiated from constraint types in order to validate both metadata and data represented by any vocabulary. As constraint types are used to define constraints on (meta)data expressed by any vocabulary, the proposed constraint classification can be applied generically, i.e. vocabulary-independent. Within the context of a complex and complete real world running example from the *SBE* domain, we prove the claim that the developed classification system perfectly applies for diverse vocabularies. We describe why RDF validation is important for the *SBE* community (section 2), show how data in rectangular format (*PHDD*) and metadata on person-level data sets (*Disco*), aggregated data sets (*QB*), thesauri (*SKOS*), and statistical classifications (*XKOS*) are represented in RDF and how therefore reused vocabularies are interrelated (section 3). We explain how *SBE* (meta)data is validated against constraints, instantiated from constraint types organized in the constraint classification system, to ensure high quality of and trust in (meta)data.

For the extensive evaluation, we implemented 53 constraint types by instantiating 212 constraints on *Disco*, *QB*, *SKOS*, *XKOS*, and *PHDD* data sets (section 5). First, several *SBE* domain experts evaluated the correctness (i.e., the gold standard) of all constraints and therefore the generic applicability of the developed constraint classification system. Second, we exhaustively evaluated the metadata quality of large real world aggregated (*QB*), person-level (*Disco*), and thesauri (*SKOS*) data sets (more than 4.2 billion triples and 15 thousand data sets) by means of constraints of the majority of the constraint types (section 6).

2 Motivation

The data most often used in research within the SBE community is *person-level data*, i.e. data collected about individuals, businesses, and households in form of responses to studies or taken from administrative registers (such as hospital records, registers of births and deaths). The range of person-level data covers many different domains and is very broad - including census, education, and health data as well as all types of business, social, and labor force surveys. Increasingly, this type of research data is held within data archives or data libraries after it has been collected, so that it may be reused by future researchers. In performing their research, the detailed person-level data is aggregated into less confidential multi-dimensional tables which answer particular research questions. Portals harvest metadata (as well as publicly available data) from multiple data providers in form of RDF. To ensure high quality, the metadata must satisfy certain criteria - specified in terms of RDF constraints. After validating the metadata according to these constraints, portals offer added values to their customers, e.g. by searching over and comparing metadata of multiple providers.

Kai: @KAI: Dieses Kapitel werde ich noch deutlich kürzen!

By its nature, person-level data is highly confidential and access is often only permitted for qualified researchers who must apply for access. The purpose of publicly available aggregated data, on the other hand, is to get a first overview and to gain an interest in further analyses on the underlying person-level data.

Researchers typically represent their results as aggregated data in form of two-dimensional tables with only a few columns (so-called *variables* such as *sex* or *age*). The *RDF Data Cube Vocabulary (QB)*[11] is a W3C recommendation for representing *data cubes*, i.e. multi-dimensional aggregate data, in RDF [6]. Aggregate data is derived from person-level data by statistics on groups or aggregates such as counts, means, and frequencies. The SDMX metadata standard – used as the basis for *QB* – and DDI have traditionally made efforts to align their content. Similarly, some of the developers of *Disco* were also involved in the development of *QB*, allowing the RDF versions of these standards to retain that alignment. While *Disco* and *QB* provide terms for the description of data sets, both on a different level of aggregation, the *Data Catalog Vocabulary (DCAT)*[12] enables the representation of these data sets inside of data collections like repositories, catalogs, or archives. The relationship between data collections and their contained data sets is useful, since such collections are a typical entry point when searching for data. Although, in most cases aggregated data is still published in form of PDFs, it is more and more common to publish aggregated data as CSV files, allowing to perform first calculations (either using all variables or only a subset). In 2014, SBE and Linked Data community members developed the *Physical Data Description (PHDD)*[13] vocabulary to represent aggregated and person-level data in a rectangular format. The data could be either represented in records with character-separated values (CSV) or in records with fixed length.

For more detailed analyses, researchers refer to person-level data from which aggregated data is derived from, as person-level data include additional variables needed for further research. One very common example for detailed analyses on person-level data is the content-driven comparison of multiple studies. Researchers get promising findings (in form of published tables with a few columns) within a metadata portal leading to subsequent research questions like 'How to compare the unemployment rate of different countries (e.g. Germany, UK, and France) in the last 10 years grouped by age?'. The first step is to determine in which countries the unemployment rate is collected and which other variables of each country-specific study are theoretically comparable and can therefore be used to answer the underlying research question. A *study* represents the process by which a data set was generated or collected. Variables are constructed out of values (of one or multiple datatypes) and/or code lists. The variable *age*, e.g., may be represented by values of the datatype *xsd:nonNegativeInteger*, or by a code list including multiple age clusters (such as '0 to 10' and '11 to 20'). To determine if variables measuring *age* - collected within multiple studies of different countries (age_{DE}, age_{UK}) - are comparable, both content-driven and technology-driven validation is performed. An example for a content-driven validation is to investigate if variables are represented in a compatible way, i.e. are the variables' code lists theoretically comparable. Technically, it can be validated (1) if vari-

[11] http://www.w3.org/TR/vocab-data-cube/
[12] http://www.w3.org/TR/vocab-dcat/
[13] https://github.com/linked-statistics/physical-data-description

able definitions are available, (2) if code lists are properly structured, and (3) if for each code an associated category (a human-readable label) is specified.

Data providers and harvesters do not only offer metadata but also publicly available data on different level of detail. To ensure high data quality and trust, they have to analyze and validate the data (are fundamental data fragments available?, how does valid data look like?). Provenance (where does the data come from?) is an important aspect in evaluating data quality. As data searchers know exactly which data sources they trust and which are reasonable to meet their individual use cases, RDF data validation can only be performed semi-automatically, i.e., an automatic approach serves as basis for intellectual decisions.

3 Vocabularies to Represent Metadata and Data in RDF

Kai: @KAI: Dieses Kapitel werde ich noch deutlich kürzen!

Eurostat[14] is the statistical office of the European Union. Its task is to provide statistics at European level that enable comparisons between countries and regions. Eurostat provides publicly available European aggregated data (downloadable in machine-readable format like CSV files) and its metadata (only textual descriptions). SBE researchers have a strong interest, e.g., in the availability of childcare services across European Union Member States.

Thomas: figure / metadata: person-level, aggregated, thesauri, catalogs, statistical classifications / data: rectangular data

Thomas: folgendes beispiel bezieht sich auf diesen datensatz

Metadata on Aggregated Data. The variable *formal childcare*[15] (in contrast to childcare at home) captures the measured availability of childcare services in percent over the population. The present data collection refers to data on formal childcare and other types of care by the variables *year*, *duration* (0 hours, 1 - 29 HPW; 30 HPW), *age* of the child (0-2 years; 3 to admission age for compulsory school; admission age for compulsory school to 12) and *country*. *QB* is a vocabulary to represent metadata on multi-dimensional aggregate data. Such data is represented using *QB* in two files, a *qb:DataSet* and a *qb:DataStructureDefinition*. The *qb:DataStructureDefinition* contains metadata of the present data collection. Thereby, the variable *formal childcare* is modelled as *qb:measure*, since it expresses what has been measured in the data collection. The *year*, *duration*, *age*, etc. are defined as *qb:dimension*. The particular data values, i.e. the specific availability of childcare services in percent over the population in particular years, are collected in a *qb:DataSet*. Each data value is represented inside a *qb:Observation* which additionally contains the particular dimensional values associated with that data value, e.g. the concrete year in which the formal childcare has been determined.

[16]

Thomas: laut SDMX sind year, duration, age, etc. keine Variablen, sondern Dimensionen

Rectangular Data. As Eurostat provides aggregated data as CSV files, we can easily represent the two-dimensional table in RDF by means of *PHDD*.

[14] http://ec.europa.eu/eurostat

[15] http://ec.europa.eu/eurostat/web/products-datasets/-/ilc_caindformal

[16] The complete running example in RDF is available at: XXXXX

PHDD is a vocabulary to represent aggregated and person-level data in a rectangular format. The data could be either represented in records with character-separated values (CSV) or in records with fixed length. The two-dimensional table about *formal childcare* can be downloaded as CSV file. The table is represented as *phdd:Table* and structured by a table structure (*phdd:TableStructure*, *phdd:Delimited*). The running example table structure includes information about the character set (*ASCII*), the variable delimiter (*,*), the new line marker (*CRLF*), and the first line where the data starts (*2*). The table structure relates to the table columns (*phdd:Column*) which are described by column descriptions (*phdd:DelimitedColumnDescription*). For the column containing the cell values in percent, e.g., the CSV column position (*5*), the recommended data type (*xsd:nonNegativeInteger*), and the storage format (*TINYINT*) is stated. The RDFication enables further aggregations and calculations, e.g., in order to compare *formal childcare* between Northern and Southern Europe or between otherwise grouped countries.

Metadata on Person-Level Data. For a broader view of the data framework and more detailed analyses we refer to the metadata on person-level data collected for the series *EU-SILC (European Union Statistics on Income and Living Conditions)*[17] and publicly provided by the *Microdata Information System (MISSY)*[18]. Where data collection is cyclic, data sets may be released as a *series*, where each cycle of the data collection activity produces one or more data sets. *Missy* is an online service platform that provides systematically structured metadata for official statistics on European person-level data sets. This includes data documentation at the study and variable level as well as documentation materials, tools and further information. Aggregated (qb:DataSet) and underlying person-level data sets (*disco:LogicalDataSet*) are connected by *prov:wasDerivedFrom*. The aggregated variable *formal childcare* is calculated on the basis of six person-level variables like *Education at pre-school*[19]. For each person-level variable detailed metadata is given (definitions and descriptions, theoretical concepts, underlying questions, code lists, frequencies and descriptive statistics, countries, year of data collection, classifications) enabling researchers to replicate the results shown in the aggregated data tables from Eurostat. Metadata on person-level data is represented in RDF using the *Disco* vocabulary. The series (*disco:StudyGroup*) *EU-SILC* contains (*disco:inGroup*) one study (*disco:Study*) for each year (*dcterms:temporal*) of data collection, e.g. *EU-SILC 2011*. The property *dcterms:spatial* points to the countries (*dcterms:Location* resources which are the same as *GeoNames* resources representing these countries) for which the data has been collected. The study *EU-SILC 2011* contains (*disco:product*) eight person-level data sets (*disco:LogicalDataSet*). Data sets include (*disco:variable*) person-level variables (*disco:Variable*) like the six ones needed to calculate the aggregated variable *formal childcare*. Metadata on

[17] http://www.gesis.org/missy/eu/metadata/EU-SILC
[18] http://www.gesis.org/missy/eu/missy-home
[19] http://www.gesis.org/missy/eu/metadata/EU-SILC/2011/Cross-sectional/original#2011-Cross-sectional-RL010

person-level data enables researchers to investigate further research questions based on promising findings of other researchers in form of aggregated data. One common research question is ,e.g., the comparison of variables like *formal childcare* between countries, for which the variable is collected within the context of an individual study, and other European or non European countries (e.g. OSCE).

Organizations, Hierarchies, and Classifications. The *Simple Knowledge Organization System (SKOS)* is used multiple times within the context of aggregated and person-level (meta)data. Variables, e.g., are constructed (*disco:representation*) out of values (of one or multiple datatypes) and/or code lists. The values of the variable *Education at pre-school*, representing the number of education hours during a usual week, are expressed as *skos:Concepts*. *Disco* uses *skos:OrderedCollection* to organize them in a particular order in a *skos:memberList*. SKOS is also used to form hierarchies of *SBE theoretical concepts* (e.g. Education) with which variables may be associated. The compete hierarchy (*skos:ConceptScheme*) of theoretical concepts (*skos:Concepts*) of series is built using *skos:narrower*. The variable *Education at pre-school* is assigned to the theoretical concept *Child Care* which is a narrower concept of *Education* - one of the top concept of the series *EU-SILC*. Controlled vocabularies serve as extension and reuse mechanism. For *Disco*, concepts (*skos:Concepts*), organized within controlled vocabularies (*skos:ConceptSchemes*), indicate types of descriptive statistics (disco:SummaryStatistics) like minimum, maximum, mean, and standard deviation. From 2012 to 2015, SBE and Linked Data community members developed *XKOS*[20] - a SKOS extension to describe formal statistical classifications like the International Standard Classification of Occupations (*ISCO*).

Searching for (Meta)data. *DCAT* enables to represent aggregated and person-level data inside of data collections like portals, repositories, catalogs, or archives serving as typical entry points when searching for data. Users search for aggregated and person-level data records (*dcat:CatalogRecord*) inside data catalogs (*dcat:Catalog*). This search differs depending on the users' information need. While it is possible to search for metadata provided inside such a record (e.g. *dcterms:title*, *dcterms:description*), users can also formulate more sophisticated queries on aggregated and person-level data sets (*dcat:Dataset*) or their distributions (*dcat:Distribution*), which are part of the records. Users may want to search for data sets covering particular topical (*dcat:keyword*, *dcat:theme*), temporal (*dcterms:temporal*), or spatial coverages (*dcterms:spatial*), or certain formats in which the data distribution is available (*dcterms:format*).

4 RDF Constraints Classification

Bosch et al. identified 74 requirements to formulate RDF constraints (e.g. *R-75, R-81: minimum qualified cardinality restrictions*); each of them corresponding

[20] https://github.com/linked-statistics/xkos

to an RDF constraint type[21][3]. We published a technical report[22] in which we explain each requirement (constraint type) in detail and give examples for each expressed by different constraint languages. The knowledge representation formalism *Description logics (DL)*, with its well-studied theoretical properties, provides the foundational basis for each constraint type. Therefore, this technical report contains mappings to DL to logically underpin each requirement and to determine which DL constructs are needed to express each constraint type [3].

We developed a system to classify RDF constraints and RDF constraint types. Constraints are instantiated from constraint types in order to validate both metadata and data represented by any vocabulary; thus, the proposed classification system is vocabulary-independent and therefore applicable generically. The complete set of *constraint types* ($\mathcal{CT}$) encompasses two disjoint ***sets of constraint types***:

1. $\mathcal{CT}_C$: *Content-Driven Constraint Types*
2. $\mathcal{CT}_T$: *Technology-Driven Constraint Types*

$\mathcal{CT}_C$ (*Content-Driven Constraint Types*) is the set of constraints ensuring that the data is consistent with the intended syntax, semantics, and integrity of vocabularies' data models (section 4.1). Therefore, domain experts may formulate $\mathcal{CT}_C$ constraints manually in prose English which is technically expressed by a constraint language afterwards. For assessing the quality of thesauri, e.g., we concentrate on the graph-based structure and apply graph- and network-analysis techniques (constraint type: *structure*). A thesaurus, e.g., should not contain many orphan concepts (concepts without any associative or hierarchical relations) lacking valuable context information for search and retrieval. $\mathcal{CT}_T$ (*Technology-Driven Constraint Types*) is the set of constraints which can be generated completely automatically out of vocabularies' data models (section 4.2). With *minimum qualified cardinality restrictions* (*R-74*), e.g., one can restrict that a *phdd:TableStructure* has (*phdd:column*) at least one *phdd:Column* (in DL: $\texttt{TableStructure} \sqsubseteq \geqslant 1\ \texttt{column.Column}$)[23].

We determined the default ***severity level*** (corresponds to requirement *R-158*) for each constraint to indicate how serious the violation of the constraint is. We propose an extensible metric to measure the continuum of severity levels ranging from $\mathcal{SL}_0$ to $\mathcal{SL}_2$. According to the constraints' default severity level the complete set of constraints ($\mathcal{C}$) encompasses three disjoint ***sets of constraints***:

1. $\mathcal{SL}_0$: set of constraints with severity level *informational*
2. $\mathcal{SL}_1$: set of constraints with severity level *warning*
3. $\mathcal{SL}_2$: set of constraints with severity level *error*

Violations of $\mathcal{SL}_0$ constraints point to possible data improvements to achieve RDF representations which are ideal in terms of syntax and semantics of used

[21] Constraint types and constraints are uniquely identified by alphanumeric technical identifiers like *R-71-CONDITIONAL-PROPERTIES*

[22] Available at: http://arxiv.org/abs/1501.03933

[23] For simplicity reasons, we do not use namespace prefixes in DL statements.

vocabularies. Data not conforming to $\mathcal{SL}_1$ and $\mathcal{SL}_2$ constraints is syntactically or semantically not correctly represented. The difference between $\mathcal{SL}_1$ and $\mathcal{SL}_2$ constraints is that $\mathcal{SL}_1$ invalid data could be whereas $\mathcal{SL}_1$ invalid data cannot be processed further. Although, we provide default severity levels for each constraint, users should be able to specify constraints' severity levels according to their individual needs, i.e., use case specific severity levels.

We recently published a technical report[24] (serving as first appendix of this paper) in which we describe and classify constraints to validate rectangular data and metadata on person-level data sets, aggregated data sets, thesauri, and statistical classifications and therefore apply the proposed classification system [5]. In this section, we describe constraint types and constraints which are important to ensure (meta)data quality on different aggregation levels. Furthermore, we associate constraints with default severity levels and assign them to validation types.

4.1 Content-Driven Constraint Types

Kai: @KAI: Dieses Kapitel werde ich noch umorganisieren! Die einzelnen Paragraphen sollten zusammengehörigen Clustern von constraint types entsprechen!

Thomas: short description

Observations of Aggregated Data Sets. The purpose of some constraints is to ensure the integrity of the data according to intended data model semantics (*data model consistency*). Every *qb:Observation*, e.g., must have a value for each dimension declared in its associated *qb:DataStructureDefinition* ($\mathcal{SL}_2$) and no two *qb:Observations* in the same *qb:DataSet* may have the same value for all dimensions ($\mathcal{SL}_1$). If a *qb:DataSet* D has a *qb:Slice* S, and S has an *qb:Observation* O, then the *qb:DataSet* corresponding to O must be D ($\mathcal{SL}_1$).

Series, Studies, Data Sets, and Data Files. It is useful to declare properties to be *conditional* (*R-71*), i.e., if particular properties exist (or do not exist), then other properties must also be present (or absent). To get an overview over a series/study either an abstract, a title, an alternative title, or links to external descriptions should be provided. If an abstract and an external description are absent, a title or an alternative title has to be stated ($\mathcal{SL}_1$). For datatype properties it should be possible to declare frequently needed *facets* (*R-46*) to drive user interfaces and validate input against simple conditions including min/max values, regular expressions, and string length. The abstract of series/studies, e.g., should have a minimum length (*xsd:minLength*; $\mathcal{SL}_1$). *Existential quantifications* (*R-86*) enforce that instances of given classes must have some property relation to individuals of certain types. If a study, e.g., has no associated data sets ($\mathcal{SL}_2$), the actual description of the data is missing which may indicate that it is very hard or unlikely to get access to the data. If there is no metadata on data files including the actual data of data sets (especially case and variable quantities; $\mathcal{SL}_1$), the description of the data sets and the underlying study is not sufficient. The case quantity measures how many cases are collected for a study.

[24] Available at: http://arxiv.org/abs/XXXXX

A high case and variable quantity is an indicator for high statistical quality and comprehensiveness of the conducted study ($\mathcal{SL}_1$).

Variables and Variable Comparison. Each variable should have a variable representation (*R-86*; $\mathcal{SL}_1$), which is either an (un)ordered code list or a union of datatypes. In case of a code list, associated categories (human-readable labels) may be stated (*R-71*; $\mathcal{SL}_0$). The variable *Education at pre-school* is represented as ordered code list without any categories. If a *skos:Concept* stands for a code (having a *skos:notation* property) and a category (having a *skos:prefLabel* property), then the property *disco:isValid* has to be stated indicating if the code is valid (*true*) or missing (*false*) (*R-71*; $\mathcal{SL}_2$). Variables may have at least one relationship to a theoretical concept (*R-86*; $\mathcal{SL}_0$). The variable *Education at pre-school* is associated with the theoretical concept *Child Care*. The default severity level of this constraint is weak, as in most cases research can be continued without associated theoretical concepts. A very common research question is to compare variables of multiple studies or countries (*comparison*). To compare variables, (1) variables and (2) variable definitions must be present, (3) code lists must be structured properly, (4) for each code an associated category (human-readable label) must be specified, and (5) code lists must either be identical or at least similar. If a researcher wants to get a first overview over comparable variables (use case 1), covering the first three constraints may be sufficient for this purpose. Thus, the severity level of the first three constraints is stronger ($\mathcal{SL}_2$) than the severity level of the next two constraints ($\mathcal{SL}_1$ and $\mathcal{SL}_0$). If the intention of the researcher is to perform more detailed comparisons (use case 2), however, the violation of the remaining two constraints is getting more serious.

Descriptive Statistics. The property *disco:percentage* stands for the number of cases of a given code in relation to the total number of cases for a particular variable within a data set. Percentage values are only valid when they are within the *literal range* of 0 and 100 (*R-45*; $\mathcal{SL}_2$). *Mathematical Operations* (*R-41, R-42*; e.g. date calculations and statistical computations like average, mean, and sum) are performed to ensure the integrity of data models. The sum of percentage values of all codes of a variable code list, e.g., must exactly be 100 ($\mathcal{SL}_2$) and the minimum of all variable codes do not have to be greater than the maximum ($\mathcal{SL}_2$). Codes (*skos:Concept*) are ordered and therefore have fixed positions in an ordered collection (*skos:OrderedCollection*) within variable representations. In order to check the correctness of relative frequencies' calculations, the cumulative percentage (*disco:cumulativePercentage*) of the current code must exactly be the cumulative percentage of the previous code plus the percentage value (*disco:percentage*) of the current code (*data model consistency*; $\mathcal{SL}_2$).

Unique Identification. It is often useful to declare a given (data) property as the *primary key* (*R-226*) of a class, so that a system can enforce uniqueness and also automatically build URIs from user inputs and imported data. In *Disco*, resources are uniquely identified by the property *adms:identifier*, which is therefore inverse-functional (`funct identifier`$^-$), i.e. for each *rdfs:Resource* x, there can be at most one distinct *rdfs:Resource* y such that y is connected by *adms:identifier*$^-$ to x ($\mathcal{SL}_2$). Keys, however, are even more general than inverse-functional properties (*R-58*), as a key can be a data, an object property, or a chain of properties [10]. For this generalization purposes, as there are different sorts of key, and as keys can lead to undecidability, DL is extended with *key boxes* and a special *keyfor* construct (`identifier keyfor Resource`) [8]. OWL 2 *hasKey* implements *keyfor* ($\mathcal{SL}_2$) and thus can be used to identify resources uniquely, to merge resources with identical key property values, and to recognize constraint violations.

Membership in Controlled Vocabularies. In many cases, resources must be members of controlled vocabularies (*R-32*). If a dimension property, e.g., has a *qb:codeList*, then the value of the dimension property on every *qb:Observation* must be in the code list ($\mathcal{SL}_2$). Summary statistics types like minimum, maximum, and arithmetic mean are maintained within a controlled vocabulary. Summary statistics can only have *disco:summaryStatisticType* relationships to *skos:Concept*s which must be members of the controlled vocabulary *ddicv:SummaryStatisticType*, a *skos:ConceptScheme* ($\mathcal{SL}_2$).

Coverage. Information about the temporal (*dcterms:temporal*), the spatial (*dcterms:spatial*), and the topical coverage (*dcterms:subject*) of series, studies, data sets, and data files (*R-86*; $\mathcal{SL}_1$) is of interest when performing frequently formulated queries (e.g. to search for all data sets of given years (temporal coverage) in which data is collected in certain countries (spatial coverage) about particular topics (topical coverage)). Depending on property datatypes, two different literal values have a specific ordering with respect to an operator like $<$(*R-43: literal value comparison*). Start dates (*disco:startDate*), e.g., must be before ($<$) end dates (*disco:endDate*) ($\mathcal{SL}_2$).

Hierarchies and Ordering. SKOS is based on RDF, which is a graph-based data model. Therefore, we can concentrate on the vocabulary's graph-based structure for assessing the quality of SKOS vocabularies and apply graph- and network-analysis techniques (*structure*) like (1) a vocabulary should provide entry points (top concepts) to the data to provide efficient access and guidance for human users, (2) concepts, internal to the tree, should not be indicated as top concepts, and (3) a vocabulary should not contain many orphan concepts (concepts without any associative or hierarchical relations) lacking valuable context information. A controlled vocabulary that contains many orphan concepts is less usable for search and retrieval use cases. Objects and literals can be *ordered* (*R-121, R-217*) for given properties. *Disco* variables, questions, and codes/categories are typically organized in a particular order. If a variable code list should be ordered, the variable representation should be of the type *skos:OrderedCollection* containing multiple codes/categories (each represented as *skos:Concept*) in a *skos:memberList*.

Reusability. Within the context of *Disco*, *skos:Concept*s can have either *skos:definition* (when interpreted as theoretical concepts) or *skos:notation* and *skos:prefLabel* properties (when interpreted as codes/categories), but not both ($\mathcal{SL}_2$). The constraint type *context-specific exclusive or of property groups* (*R-11*) restricts individuals of given classes to have exactly one of multiple property groups.

4.2 Technology-Driven Constraint Types

Constraints of some constraint types are directly and automatically derived from syntax and semantics of vocabularies' conceptual models. As these constraints depend on data models' intended semantics, associated default severity levels are in most cases very strong ($\mathcal{SL}_2$).

Vocabulary. One should not invent new or use deprecated terms of vocabularies (*vocabulary*). *Property Domains* (*R-25, R-26*) and *Ranges* (*R-28, R-35*) restrict domains and ranges of properties. Only *phdd:Tables*, e.g., can have *phdd:isStructuredBy* relationships ($\exists$ `isStructuredBy.`$\top \sqsubseteq$ `Table`) and *xkos:belongsTo* relationships can only point to *skos:Concept* instances ($\top \sqsubseteq \forall$ `belongsTo.Concept`). A *universal quantification* (*R-91*) contains all those individuals that are connected by a property only to individuals/literals of particular classes or data ranges. Only *dcat:Catalogs*, e.g., can have *dcat:dataset* relationships to *dcat:Datasets* (`Catalog` $\sqsubseteq \forall$ `dataset.Dataset`). Out-dated classes and properties of previous vocabulary versions can be marked as deprecated. The constraint type *context-specific valid classes and properties* (*R-209; R-210*) can be used to specify which classes and properties are valid in which context - here a given vocabulary version. Many properties are not necessarily required but *recommended* within a particular context (*R-72*). The property *skos:notation*, e.g., is not mandatory for *disco:Variable*s, but recommended to represent variable names. *R-223* serves to make sure that all literal values are valid with regard to their datatypes. Thus, all date values (e.g. *disco:startDate*, *disco:endDate*, *dcterms:date*) must be of the datatype *xsd:date* and *xsd:nonNegativeInteger* values (e.g. *disco:frequency*) do not have to be negative.

Cardinality Restrictions. An *existential quantification* (*R-86*) contains all those individuals that are connected by a property to individuals/literals of given classes or data ranges. Every *qb:SliceKey*, e.g., must be associated with (*qb:sliceKey*) a *qb:DataStructureDefinition* (`SliceKey` $\sqsubseteq \exists$ `sliceKey`$^{-}$`.DataStructureDefinition`). *Minimum/maximum/exact qualified cardinality restrictions* (*R-74, R-75, R-76*) contain all those individuals that are connected by a property to at least/at most/exactly n different individuals/literals of particular classes or data ranges. A *phdd:TableStructure*, e.g., has at least one *phdd:column* relationship to a *phdd:Column* (`TableStructure` $\sqsubseteq \geqslant$`1 column.Column`), a *disco:Variable* has at most one *disco:concept* relationship to a theoretical concept (*skos:Concept*) (`Variable` $\sqsubseteq \leqslant$`1 concept.Concept`), and a *qb:DataSet* has (*qb:structure*) exactly one associated *qb:DataStructureDefinition* (`DataSet` $\sqsubseteq \geqslant$`1 structure.DataStructureDefinition` $\sqcap \leqslant$`1 structure.DataStructureDefinition`).

Language Tag Cardinality. For data properties, it may be desirable to restrict that values of predefined languages must be present for determined number of times (*R-48, R-49*): (1) Some controlled vocabularies contain literals in natural language, but without information what language has actually been used. (2) Language tags must conform to language standards. (3) Some thesaurus concepts are labeled in only one, others in multiple languages. It may be desirable to have each concept labeled in each of the languages that are also used on the other concepts. Although not always possible, incompleteness of language coverage for some concepts may indicate shortcomings of thesauri [9].

Disjointness and Allowed Values. All properties, not having the same domain and range classes, are defined to be pairwise disjoint (*R-9: disjoint properties*), stating that no individual x can be connected to an individual/literal y by disjoint properties like *phdd:isStructuredBy* and *phdd:column* ($isStructuredBy \sqsubseteq \neg column$). All *XKOS* classes are pairwise disjoint (*R-7: disjoint classes*; e.g. $\texttt{ClassificationLevel} \sqcap \texttt{ConceptAssociation} \sqsubseteq \bot$), i.e., individuals cannot be instances of multiple disjoint classes. It is a common requirement to narrow down the value space of properties by an exhaustive enumeration of valid values. *Allowed values* (*R-30, R-37*) for properties can be IRIs (matching one or multiple patterns), any literals, allowed literals (e.g. 'red' 'blue' 'green'), and typed literals of one or multiple type(s) (e.g. *xsd:string*) - *disco:CategoryStatistics*, e.g., can only have *disco:computationBase* relationships to the values *valid* and *invalid* of the datatype *rdf:langString* ($\texttt{CategoryStatistics} \equiv \forall\ \texttt{computationBase.}\{\texttt{valid,invalid}\} \sqcap \texttt{langString}$).

Validation and Reasoning. Some constraint types enable performing reasoning prior to validation which may resolve or cause constraint violations. With *subsumption* (*R-100*) one can state that *xkos:ClassificationLevel* is a subclass of *skos:Collection*, i.e., each *xkos:ClassificationLevel* resource must also be part of the *skos:Collection* class extension ($\texttt{xkos:ClassificationLevel} \sqsubseteq \texttt{skos:Collection}$). With *sub properties* (*R-54, R-64*) one can state that *disco:fundedBy* is a sub-property of *dcterms:contributor* - i.e., if a study is funded by an organization, then this organization contributed to this study ($\texttt{disco:fundedBy} \sqsubseteq \texttt{dcterms:contributor}$). *Asymmetric object properties* (*R-62*) restrict that if individual x is connected by the object property OP to individual y, then y cannot be connected by OP to x. Such constraints are defined for each object property for which a semantically equivalent object property pointing from the other direction would also be possible but is not defined within the vocabulary. A *disco:Variable*, e.g., may be based on (*disco:basedOn*) a *disco:RepresentedVariable*. A *disco:RepresentedVariable*, however, cannot be based on a *disco:Variable* ($disco : basedOn \sqcap disco : basedOn^{-} \sqsubseteq \bot$). *Default values* (*R-31, R-38*) for objects/literals of given prooperties are inferred automatically when properties are not present. The value *true* for the property *disco:isPublic* indicates that a *disco:LogicalDataSet* can be accessed by anyone. Per default, however, access to data sets should be restricted (*false*). Validation should *exploit sub-super relations* in vocabularies (*R-224*). If *dcterms:coverage* and one of its sub-properties (*dcterms:spatial*, *dcterms:temporal*) are present, it is checked that *dcterms:coverage* is not redundant with its sub-properties. This validation can indicate when the data is verbose (redundant) or expressed at a too general level and could thus be improved.

5 Implementation

SPARQL is generally seen as the method of choice to validate RDF data according to certain constraints. We use *SPIN*, a SPARQL-based way to formulate and check constraints, as basis to develop a validation environment (available at http://purl.org/net/rdfval-demo)[25] to validate RDF data according to constraints expressed my arbitrary constraint languages like Shape Expressions, Resource Shapes, and the Web Ontology Language[26] [2]. The *RDF Validator* also validates RDF data to ensure correct syntax, semantics, and integrity of diverse vocabularies such as *Disco, QB, PHDD, SKOS, and XKOS*. Although accessible within our validation tool, we provide all implemented constraints[27] in form of SPARQL CONSTRUCT queries. For the subsequent evaluation, we implemented 212 constraints on *Disco*, *QB*, *SKOS*, *XKOS*, and *PHDD* data sets. The SPIN engine checks for each resource if it satisfies all constraints, which are associated with its assigned classes, and generates a result RDF graph containing information about all constraint violations. There is one SPIN construct template for each constraint type and vocabulary-specific constraint[28]. A SPIN construct template contains a SPARQL CONSTRUCT query which generates constraint violation triples indicating the subject and the properties causing constraint violations, and the reason why constraint violations have been raised. A SPIN construct template creates constraint violation triples if all triple patterns within the SPARQL WHERE clause match. *Missy*[29] provides comprehensive Linked Data services like diverse RDF exports of person-level metadata conforming to the *Disco* vocabulary in form of multiple concrete syntaxes.

6 Evaluation

6.1 Evaluation Setup

First, several SBE domain experts of the vocabularies *Disco*, *QB*, *SKOS*, *XKOS*, and *PHDD* evaluated the correctness (i.e., the gold standard) of all $\mathcal{CT}_C$ and $\mathcal{CT}_T$ constraints and therefore the generic applicability of the developed classification system of constraint types and constraints. Second, we exhaustively evaluated the metadata quality of large real world aggregated (*QB*), person-level (*Disco*), and thesauri (*SKOS*) data sets by means of both $\mathcal{C}_C$ and $\mathcal{C}_T$ constraints of the majority of the constraint types. We validated 9,990 / 3,775,983,610 (*QB*), 4,178 / 477,737,281 (*SKOS*), and 1,526 / 9,673,055 (*Disco*) data sets / triples using the *RDF Validator* in batch mode. That are more than 4.2 billion triples and 15 thousand data sets. We validated, i.a., (1) *QB* data sets published by the *Australian Bureau of Statistics (ABS)*, the *European Central Bank (ECB)*, and the *Organisation for Economic Co-operation and Development (OECD)*, (2) *SKOS* thesauri like the *AGROVOC Multilingual agricultural thesaurus*, the *STW Thesaurus for Economics*, and the *Thesaurus for the Social Sciences (TheSoz)*, and (3) *Disco* data sets provided by the *Microdata Information System (Missy)*,

[25] Source code downloadable at: https://github.com/boschthomas/rdf-validator
[26] SPIN mappings available at: https://github.com/boschthomas/rdf-validation/SPIN
[27] https://github.com/boschthomas/rdf-validation/tree/master/constraints
[28] For a comprehensive description of the *RDF Validator*, we refer to [2]
[29] http://www.gesis.org/missy/eu/missy-home

the *DwB Discovery Portal*, the *Danish Data Archive (DDA)*, and the *Swedish National Data Service (SND)*.

We recently published a technical report[30] (serving as second appendix of this paper) in which we describe the comprehensive evaluation in detail [4]. As we evaluated nearly 10 thousand *QB* data sets, we published the evaluation results for each data set in form of one document per SPARQL endpoint[31]. Table 1 shows the evaluation results.

6.2 Evaluation Results and Discussion

Criteria	***Disco***	***QB***	***SKOS***	**Total**
Triples	9,673,055	3,775,983,610	477,737,281	4,263,393,946
Data Sets	1,526	9,990	4,178	15,694
CV	3,545,703	45,635,846	5,540,988	54,722,537
CV ($\mathcal{SL}_0$)	2,437,922 (**68.8%**)	0 (0%)	2,281,740 (41.2%)	4,719,662 (8.6%)
CV ($\mathcal{SL}_1$)	473,574 (13.4%)	45,520,613 (**99.75%**)	3,259,248 (**58.8%**)	49,253,435 (**90%**)
CV ($\mathcal{SL}_2$)	634,207 (17.9%)	115,233 (0.25%)	0 (0%)	749,440 (1.4%)
CT	52	20	14	53
CT ($\mathcal{C}_C$)	30 (**57.7%**)	5 (25%)	5 (35.7%)	30 (**56.6%**)
CT ($\mathcal{C}_T$)	22 (42.3%)	15 (**75%**)	9 (**64.3%**)	23 (43.4%)
C	142	35	35	212
C ($\mathcal{C}_C$)	72 (**50.7%**)	16 (45.7%)	21 (**60%**)	109 (**51.4%**)
C ($\mathcal{C}_T$)	70 (**49.3%**)	19 (**54.3%**)	14 (40%)	103 (**48.6%**)
C ($\mathcal{SL}_0$)	75 (**52.8%**)	4 (11.4%)	21 (**60%**)	100 (**47.2%**)
C ($\mathcal{SL}_1$)	9 (6.3%)	3 (8.6%)	5 (14.3%)	17 (8%)
C ($\mathcal{SL}_2$)	58 (40.8%)	28 (**80%**)	9 (25.7%)	95 (**44.8%**)

C (constraints), CT (constraint types), CV (constraint violations)

Table 1: Evaluation

We identified 142 *Disco* constraints ($\mathcal{C}_C$ and $\mathcal{C}_T$ constraints to the same extend) assigned to 52 distinct constraint types and implemented 77 of them to actually validate person-level data sets. For *QB*, we specified more $\mathcal{C}_T$ (54%) than $\mathcal{C}_C$ constraints; for *SKOS*, however, more $\mathcal{C}_C$ constraints (60%). We instantiated more $\mathcal{C}_C$ (58%) than $\mathcal{C}_T$ constraint types to define *Disco* constraints; for *QB* (75%) and *SKOS* (64%), on the other side, more $\mathcal{C}_T$ constraint types. In total, we used 53 of overall 82 distinct constraint types (57% of them are $\mathcal{C}_C$ constraint types) to define 212 constraints (equally $\mathcal{C}_C$ and $\mathcal{C}_T$ constraints).

For *Disco* and *SKOS*, more than the half of the constraints are associated with the weakest severity level $\mathcal{SL}_0$. Within the context of *QB*, 80% of the constraints are classified as the most serious ones ($\mathcal{SL}_2$). All in all, there are a

[30] Available at: http://arxiv.org/abs/XXXXX

[31] Available at: https://github.com/boschthomas/rdf-validation/tree/master/evaluation/data-sets/data-cube

little bit more $\mathcal{SL}_0$ then $\mathcal{SL}_2$ constraints, whereas $\mathcal{SL}_1$ constraints are negligible. *Existential quantifications* (32.4%, *Disco*), *data model consistency* (31.4%, *QB*), and *structure* (28.6%, *SKOS*) are the constraint types the most constraints are instantiated from. By validating *QB* data sets, we got the most constraint violations (more than 45 millions), followed by *SKOS* and *Disco* (with more than 5.5 and 3.5 millions) - consequently, almost 55 million constraint violations were raised during the evaluation which could be used to enhance the metadata quality of these data sets. Close to 70% of all *Disco* constraint violations are caused by violating $\mathcal{SL}_0$ constraints. For *QB* (nearly 100%) and *SKOS* (almost 60%), the majority of the raised constraint violations are classified to be more serious ($\mathcal{SL}_1$). 80% of all *QB* constraints are $\mathcal{SL}_2$ constraints leading to less than 1% of all *QB* constraint violations. Altogether, exactly 90% of the constraint violations are assigned to the severity level $\mathcal{SL}_1$. These findings are surprising as only 8% of all defined constraints are $\mathcal{SL}_1$ constraints. The constraints responsible for the largest numbers of constraint violations are *DISCO-C-LABELING-AND-DOCUMENTATION-06* and *DISCO-C-COMPARISON-VARIABLES-02* (both 547,916) (*Disco*), *DATA-CUBE-C-DATA-MODEL-CONSISTENCY-05* (45,514,102) (*QB*), and *SKOS-C-LANGUAGE-TAG-CARDINALITY-01* (2,508,903) (*SKOS*). We refer to the technical reports[24] [30] to get details about constraints on and the evaluation of *XKOS* and *PHDD* data sets.

Thomas: kurz beschreiben

7 Related Work

For data archives, research institutes, and data libraries, RDF validation according to predefined constraints is a much sought-after feature, particularly as this is taken for granted in the XML world. DDI-XML documents, e.g., are validated against diverse XSDs[4]. As certain constraints cannot be formulated and validated by XSDs, so-called secondary-level validation tools like *Schematron*[32] have been introduced to overcome the limitations of XML validation. *Schematron* generates validation rules and validates XML documents according to them. With RDF validation, one can overcome drawbacks when validating XML documents[33]. It cannot be validated, e.g., if each code of a variable's code list is associated with a category (*R-86*). Additionally, it cannot be validated that if an element has a specific value, then certain child elements must be present (*R-71*). A comprehensive comparison of XML and RDF validation, however, is not within the scope of this paper.

A well-formed *RDF Data Cube* is an a RDF graph describing one or more instances of *qb:DataSet* for which each of the 22 integrity constraints[34], defined within the *QB* specification, passes. Each integrity constraint is expressed as narrative prose and, where possible, a SPARQL ASK query or query template.

[32] https://msdn.microsoft.com/en-us/library/aa468554.aspx
[33] http://www.xmlmind.com/xmleditor/_distrib/doc/xmltool/xsd_structure_limitations.html
[34] http://www.w3.org/TR/vocab-data-cube/#wf

If the ASK query is applied to an RDF graph then it will return true if that graph contains one or more *QB* instances which violate the corresponding constraint [7]. Mader, Haslhofer, and Isaac investigated how to support taxonomists in improving SKOS vocabularies by pointing out quality issues that go beyond the integrity constraints defined in the SKOS specification [9].

8 Conclusion and Future Work

In this paper, we showed in form of a complete real world running example how to represent metadata on person-level data (*Disco*), metadata on aggregated data (*QB*), and data on both aggregation levels in a rectangular format (*PHDD*) in RDF and how therefore used vocabularies are interrelated (**contribution 1**, section 3). We explained why RDF validation is important in this context and how metadata on person-level data, aggregated data, thesauri, and statistical classifications as well as data on both aggregation levels is validated against constraints to ensure high (meta)data quality[35] (**contribution 2**, section **??**). We distinguish two validation types: (1) *Content-Driven Validation* $\mathcal{C}_C$ contains the set of constraints ensuring that the data is consistent with the intended syntax, semantics, and integrity of data models (section 4.1). (2) *Technology-Driven Validation* $\mathcal{C}_T$ includes the set of constraints which can be generated automatically out of data models, such as cardinality restrictions, universal and existential quantifications, domains, and ranges (section 4.2). We determined the default *severity level* for each constraint to indicate how serious the violation of the constraint is and propose an extensible metric to measure the continuum of severity levels.

Kai: @KAI: Dieses Kapitel muss ich noch an die Änderungen im Paper anpassen!

We implemented a validation environment (available at http://purl.org/net/rdfval-demo) to validate RDF data according to constraints expressed my arbitrary constraint languages and to ensure correct syntax, semantics, and integrity of diverse vocabularies such as *Disco, QB, PHDD, SKOS, and XKOS* (section 5). We exhaustively evaluated the metadata quality of large real world aggregated (*QB*), person-level (*Disco*), and thesauri (*SKOS*) data sets by means of 212 $\mathcal{C}_C$ and $\mathcal{C}_T$ constraints of the majority of the constraint types. We validated more than 4.2 billion triples and 15 thousand data sets[36] (section 6).

References

1. Thomas Bosch and Kai Eckert. Requirements on rdf constraint formulation and validation. *Proceedings of the DCMI International Conference on Dublin Core and Metadata Applications (DC 2014)*, 2014.

[35] The first appendix of this paper describing each constraint in detail is available at: http://arxiv.org/abs/XXXXX [5]

[36] The second appendix of this paper describing the evaluation in detail is available at: http://arxiv.org/abs/XXXXX [4].

2. Thomas Bosch and Kai Eckert. Towards description set profiles for rdf using sparql as intermediate language. *Proceedings of the DCMI International Conference on Dublin Core and Metadata Applications (DC 2014)*, 2014.
3. Thomas Bosch, Andreas Nolle, Erman Acar, and Kai Eckert. Rdf validation requirements - evaluation and logical underpinning. 2015.
4. Thomas Bosch, Benjamin Zapilko, Joachim Wackerow, and Kai Eckert. An evaluation of metadata and data quality on person-level, aggregated, thesauri, statistical classifications, and rectangular data sets. 2015.
5. Thomas Bosch, Benjamin Zapilko, Joachim Wackerow, and Kai Eckert. Rdf constraints to validate metadata on person-level, aggregated, thesauri, and statistical classifications data sets and rectangular data. 2015.
6. Richard Cyganiak, Simon Field, Arofan Gregory, Wolfgang Halb, and Jeni Tennison. Semantic statistics: Bringing together sdmx and scovo. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *Proceedings of the WWW 2010 Workshop on Linked Data on the Web*, volume 628 of *CEUR Workshop Proceedings*, 2010.
7. Richard Cyganiak and Dave Reynolds. The rdf data cube vocabulary. W3C recommendation, W3C, January 2014.
8. Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, nominals, and concrete domains. *Journal of Artificial Intelligence Research*, 23(1):667–726, June 2005.
9. Christian Mader, Bernhard Haslhofer, and Antoine Isaac. Finding quality issues in skos vocabularies. In *Proceedings of the Second International Conference on Theory and Practice of Digital Libraries*, TPDL'12, pages 222–233, Berlin, Heidelberg, 2012. Springer-Verlag.
10. Michael Schneider. OWL 2 Web Ontology Language RDF-Based Semantics. W3C recommendation, W3C, October 2009.
11. Mary Vardigan, Pascal Heus, and Wendy Thomas. Data documentation initiative: Towards a standard for the social sciences. *International Journal of Digital Curation*, 3(1):107–113, 2008.