Informative Armstrong RDF Datasets for n-ary Relations

Henriette HARMSE [a], Katarina BRITZ [b] and Aurona GERBER [a]

[a] *CSIR CAIR, Department of Informatics, University of Pretoria, Pretoria, South Africa*

[b] *CSIR CAIR, Department of Information Science, Stellenbosch University, South Africa*

Abstract. The W3C standardized Semantic Web languages enable users to capture data without a schema in a manner which is intuitive to them. The challenge is that, for the data to be useful, it should be possible to query the data and to query it efficiently, which necessitates a schema. Understanding the structure of data is thus important to both users and storage implementers: The structure of the data gives insight to users in how to query the data while storage implementers can use the structure to optimize queries. In this paper we propose that data mining routines be used to infer candidate n-ary relations with related uniqueness- and null-free constraints, which can be used to construct an informative Armstrong RDF dataset. The benefit of an informative Armstrong RDF dataset is that it provides example data based on the original data which is a fraction of the size of the original data, while capturing the constraints of the original data faithfully. A case study on a DBPedia person dataset showed that the associated informative Armstrong RDF dataset contained 0.00003% of the statements of the original DBPedia dataset.

Keywords. informative Armstrong RDF dataset, informative Armstrong ABox, Semantic Web, data mining, example data, uniqueness constraint, n-ary relation, DBPedia

1. Introduction

In the context of the Semantic Web, applications often enable capturing of data in the absence of a schema or with limited schema information available. This is referred to as the data-first approach as opposed to the schema-first approach found in relational data theory. The benefit of this approach is that it allows users the freedom to capture information in a way that is intuitive to them. The challenge is however that, for data to be usable, it should be possible to query the data and to query it efficiently. Inferring structure is therefore useful to users since it provides guidance on how to construct queries. Moreover, understanding the structure of data is helpful to storage implementers because it gives insight into how to efficiently query the data [10].

The chosen data model of the Semantic Web is RDF, which expresses facts regarding a world in triples of the form $\langle s, p, o \rangle$, where s is the subject, p the predicate and o the object [5]. OWL 2 gives meaning to RDF data through a model-theoretic semantics, which is based on Description Logics (DLs). Description logics are syntactic variants of fragments of first-order logic that are specifically designed for the conceptual representation of an application domain in terms of concepts and relationships between concepts. A key design goal for DLs is to ensure that the core reasoning procedures are decidable. A DL ontology $\mathcal{O}$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, written as $\mathcal{O} = (\mathcal{T}, \mathcal{A})$. The TBox uses axioms to define concepts and relationships between concepts and the ABox uses assertions to assert knowledge regarding the domain of interest [6]. To enable querying of RDF data, the SPARQL Protocol and RDF Query Language (SPARQL) is defined as part of the W3C Semantic Web standards [5].

Our objective in this paper is to present algorithms for extracting an example RDF dataset from an original RDF dataset that is a small subset of the original RDF dataset. This subset of the original RDF dataset is such that for a particular n-ary relation it satisfies all uniqueness- and null-free constraints of the original RDF dataset, while violating all uniqueness- and null-free constraints that do not hold for the particular n-ary relation in the original RDF dataset. To illustrate the potential usefulness of our method we apply it to the person RDF dataset of DBPedia [1].

Intuitively, a uniqueness constraint on an n-ary relation is a subset of the n components of the relation stating that no two different elements of the relation agree on their participation in the components of the uniqueness constraint. A null-free constraint on an n-ary relation states which components of the relation will always have values (i.e. will never be null).

Our motivation for focusing on n-ary relations over which uniqueness- and null-free constraints are expressed are three-fold: Firstly, querying relational databases tends to be more efficient than querying RDF datasets, with research suggesting that storing RDF data internally as relational data can speed-up queries over RDF data [10]. Secondly, in relational databases uniqueness constraints are important for query optimization, indexing and data integration [7]. This is expected to also hold true for RDF datasets, particularly if a relational database is used internally [10]. Thirdly, a common problem experienced by users posing SPARQL queries is that the query results are empty when their assumptions w.r.t. the latent structure of RDF data are incorrect. We aim to alleviate this problem by making the n-ary relations and the null-free constraints present in RDF data explicit.

In this paper we use the notion of **informative Armstrong RDF datasets** to extract an example dataset to make the structure of data captured using a data-first approach explicit. Due to the verbose syntax of RDF, OWL 2 and SPARQL we will use the mathematical formalizations underlying these standards. Therefore we base informative Armstrong RDF datasets on **informative Armstrong ABoxes**. We assume that we start with an ABox and an empty or under-specified TBox (rather than an RDF dataset). We use conjunctive instance queries (rather than SPARQL) to determine an n-ary relation and its related uniqueness- and null-free constraints that apply to the ABox. Axioms can be added to the TBox, which makes the candidate schema underlying the ABox explicit in terms of the n-ary relation, uniqueness- and null-free constraints found. Based on the n-ary relation, uniqueness- and null-free constraints, assertions can be generated for the informative Armstrong ABox. The axioms and assertions are expressed in a combination of $\mathcal{SROIQ}^{(\mathcal{D})}$ and DL safe rules (rather than OWL 2).

An ABox $\mathcal{A}_{\mho}$ (pronounced A-shield) is an Armstrong ABox for a given n-ary relation C, with given sets of uniqueness constraints Σ and null-free constraints X, if and only if [4]:

1. all assertions of $\mathcal{A}_{\mho}$ **satisfy** each uniqueness constraint in Σ, as well as each null-free constraint in X, and
2. every uniqueness constraint not in Σ, or that cannot be derived from Σ, is **violated** by some assertion in $\mathcal{A}_{\mho}$, and every null-free constraint not in X is **violated** by some assertion in $\mathcal{A}_{\mho}$.

An informative Armstrong ABox is an **Armstrong ABox** for which the data is derived from existing data rather than being synthetically generated.

This paper is structured as follows. In Section 2 we review key definitions and results that are of importance in the development of informative Armstrong ABoxes w.r.t. Description Logics (Section 2.1) and conjunctive instance queries (Section 2.2). Section 3 presents the main contribution of this paper: (1) We give an example of an Armstrong ABox and introduce key terminology based on Harmse et al. [4] (Section 3.1). (2) We define the data mining routines for discovering n-ary relations with associated uniqueness- and null-free constraints for ABoxes (Section 3.2). (3) We define the algorithms for generating an informative Armstrong ABox (Section 3.3). (4) We validate our contribution via a case study in which we generate an informative Armstrong RDF dataset for a DBPedia dataset (Section 3.4). (5) We list the potential benefits of informative Armstrong ABoxes (Section 3.5). Section 4 concludes this paper.

2. Preliminaries

2.1. Description Logics and DL Safe Rules

The logical formalization underlying OWL 2 is based on the DL $\mathcal{SROIQ}^{(\mathcal{D})}$ [6] and DL safe rules, which is used to implement uniqueness constraints in OWL 2 [8].

The syntactic building blocks of $\mathcal{SROIQ}^{(\mathcal{D})}$ are based on the disjoint sets N_C, N_R and N_I, where N_C is a set of concept names, N_R is a set of role names and N_I is a set of individual names. $\mathcal{SROIQ}$ is a very expressive DL of which we only use a small subset of concept constructors in this paper. The constructors we use in this paper are given by:

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \forall r.C \mid \exists r.C \mid \geq nr.C \mid \leq nr.C \mid \forall t.d \mid \exists t.d \mid \geq nt.d \mid \leq nt.d$$

where A is an atomic concept, C, C_1, C_2 are concepts, d is a data type, r is a (simple) abstract role, t is a concrete role and n is an integer.

The semantics of concept descriptions is given in terms of an interpretation $\mathcal{I} = (\triangle^{\mathcal{I}}, \triangle^{\mathcal{D}}, \cdot^{\mathcal{I}})$, where $\triangle^{\mathcal{I}}$ and $\triangle^{\mathcal{D}}$ are non-empty disjoint sets such that $\triangle^{\mathcal{I}}$ is the domain of interest and $\triangle^{\mathcal{D}}$ is the domain of all data types. The function $\cdot^{\mathcal{D}}$ associates for each $d \in \mathcal{D}$, a set $d^{\mathcal{D}} \subseteq \triangle^{\mathcal{D}}$. The function $\cdot^{\mathcal{I}}$ assigns to each concept $A \in N_C$, each abstract role $r \in R_A$, each individual $a \in N_I$, and each concrete role $t \in R_D$ interpretations $A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$, $r^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$, $a^{\mathcal{I}} \in \triangle^{\mathcal{I}}$, and $t^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{D}}$ respectively. We assume $N_R = R_A \cup R_D$, $\top^{\mathcal{I}} = \triangle^{\mathcal{I}} \cup \triangle^{\mathcal{D}}$ and $\bot^{\mathcal{I}} = \varnothing$.

Given an interpretation $\mathcal{I} = (\triangle^{\mathcal{I}}, \triangle^{\mathcal{D}}, \cdot^{\mathcal{I}})$, the function $\cdot^{\mathcal{I}}$ is extended to interpret complex concepts in the following way:

$$
\begin{aligned}
(\neg C)^{\mathcal{I}} &= \triangle^{\mathcal{I}} \backslash C^{\mathcal{I}}, (C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\forall r.C)^{\mathcal{I}} &= \{x \in \triangle^{\mathcal{I}} | \text{For all } y \in \triangle^{\mathcal{I}}, \text{if } (x,y) \in r^{\mathcal{I}}, \text{ then } y \in C^{\mathcal{I}}\} \\
(\exists r.C)^{\mathcal{I}} &= \{x \in \triangle^{\mathcal{I}} | \text{There is some } y \in \triangle^{\mathcal{I}} \text{ such that } (x,y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \\
(\leq nr.C)^{\mathcal{I}} &= \{x \in \triangle^{\mathcal{I}} | \sharp\{y \in \triangle^{\mathcal{I}} | (x,y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leq n\} \\
(\forall t.d)^{\mathcal{I}} &= \{x \in \triangle^{\mathcal{I}} | \text{For all } v \in \triangle^{\mathcal{D}}, \text{ if } (x,v) \in t^{\mathcal{I}}, \text{ then } v \in d^{\mathcal{D}}\} \\
(\exists t.d)^{\mathcal{I}} &= \{x \in \triangle^{\mathcal{I}} | \text{There is some } v \in \triangle^{\mathcal{D}} \text{ exists such that } (x,v) \in r^{\mathcal{I}} \text{ and } v \in d^{\mathcal{D}}\} \\
(\leq nt.C)^{\mathcal{I}} &= \{x \in \triangle^{\mathcal{I}} | \sharp\{v \in \triangle^{\mathcal{D}} | (x,v) \in t^{\mathcal{I}} \text{ and } v \in d^{\mathcal{D}}\} \leq n\}
\end{aligned}
$$

TBox general concept inclusion (GCI) axioms are of the form $C_1 \sqsubseteq C_2$ where C_1 and C_2 are concepts. ABox assertions are of the form $C(x)$, $r(x,y)$, $t(x,v)$ or $x \not\approx y$ where C is a concept, r an abstract role, t a concrete role, x and y are individuals, and v is an element of a data type.

When an interpretation $\mathcal{I}$ **satisfies** a GCI or assertion α it is denoted by $\mathcal{I} \Vdash \alpha$. Satisfaction of α is defined as follows: $\mathcal{I} \Vdash C_1 \sqsubseteq C_2$ iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, $\mathcal{I} \Vdash C(x)$ iff $x^{\mathcal{I}} \in C^{\mathcal{I}}$, $\mathcal{I} \Vdash r(x,y)$ iff $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}}$, and $\mathcal{I} \Vdash t(x,v)$ iff $(x^{\mathcal{I}}, v^{\mathcal{D}}) \in t^{\mathcal{I}}$. $\mathcal{I}$ is a **model** of a TBox $\mathcal{T}$ or an ABox $\mathcal{A}$ if it satisfies all its GCIs or assertions. In case $\mathcal{I}$ is a model of both $\mathcal{T}$ and $\mathcal{A}$, it is also called a model of the ontology $(\mathcal{T}, \mathcal{A})$ and $(\mathcal{T}, \mathcal{A})$ is said to be **consistent** if such a model exists. An axiom or assertion α is said to be **entailed** by an ontology $\mathcal{O}$, written as $\mathcal{O} \vDash \alpha$, if every model of $\mathcal{O}$ is also a model of α.

The implementation of uniqueness constraints in OWL 2 is based on DL safe rules [8]:

$$C \texttt{ hasKey}(r_1, \ldots, r_n, t_1, \ldots, t_m)$$

which states that for any two named individuals x and y of type C it follows that $x = y$, if and only if, they agree on their participation of

1. abstract roles r_i with named individuals z_i, for $1 \leq i \leq n$, and
2. concrete roles t_i with data values v_i, for $1 \leq i \leq m$, respectively.

2.2. Conjunctive Instance Queries

The mathematical formalization underlying SPARQL is based on conjunctive instance queries, for which we provide key definitions here. Let $\mathcal{S} = (N_C, N_R, N_I)$ be a signature and $\mathcal{O}$ an ontology over $\mathcal{S}$. Let $V = V_C \cup V_R \cup V_I$ be a set that is disjoint with N_C, N_R and N_I where V_C is a set of concept variables, V_R is a set of role variables and V_I is a set of instance variables. A **concept atom** is an expression $A(\mathbf{x})$ and a **role atom** is an expression $r(\mathbf{x},\mathbf{y})$ where $A \in N_C$, $r \in N_R$ and $\mathbf{x},\mathbf{y} \in V_I$. A **conjunctive instance query** $\mathcal{Q}$ is of the form $q_1 \wedge \ldots \wedge q_n$ where $q_1,\ldots,q_n$ are concept- and/or role atoms [3].

Let $\mathcal{Q} = \{q_1,\ldots,q_n\}$ be a conjunctive instance query. $\mathsf{Var}(\mathcal{Q})$ is used to represent the set of variables in $\mathcal{Q}$. A total function $\mu : \mathsf{Var}(Q) \to N_C \cup N_R \cup N_I$ is a **mapping** for $\mathcal{Q}$ over $\mathcal{O}$ if $\mu(\mathbf{v}) \in N_C$ for $\mathbf{v} \in V_C$, $\mu(\mathbf{v}) \in N_R$ for $\mathbf{v} \in V_R$ and $\mu(\mathbf{v}) \in N_I$ for $\mathbf{v} \in V_I$. A mapping μ is an **answer** for $\mathcal{Q}$ over $\mathcal{O}$ if $\mathcal{O} \vDash \mu(q)$ for each $q \in \mathcal{Q}$, written $\mathcal{O} \vDash \mu(\mathcal{Q})$, where $\mu(q)$ ($\mu(\mathcal{Q})$) is the result of replacing each $\mathbf{v} \in \mathsf{Var}(q)$ ($\mathbf{v} \in \mathsf{Var}(\mathcal{Q})$) with $\mu(\mathbf{v})$. The set of all answers for $\mathcal{Q}$ over $\mathcal{O}$ is denoted by $\mathsf{ans}(\mathcal{O},\mathcal{Q})$.

3. Generating Informative Armstrong ABoxes

Informative Armstrong ABoxes are inspired by informative Armstrong relations in relational database theory [2]. In relational database theory the benefit of an informative Armstrong relation is that it provides a representative example that is a fraction of the size of the original relation, with some informative Armstrong relations containing 0.6% of the tuples of the original relation [7]. We expect similar reductions in size for RDF datasets.

A challenge in generating an informative Armstrong ABox is that an Armstrong ABox $\mathcal{A}_{\sqcup}$ exhibits a form of **completeness** in the sense that for an n-ary relation C with uniqueness constraints Σ and null-free constraints X[4]:

1. **every** assertion of $\mathcal{A}_{\sqcup}$ satisfies **each** uniqueness constraint in Σ and **each** null-free constraint in X, and
2. **every** uniqueness constraint not in Σ, or that cannot be derived from Σ, is violated by some assertion in $\mathcal{A}_{\sqcup}$, and **every** null-free constraint not in X, is violated by some assertion in $\mathcal{A}_{\sqcup}$.

Hence, to construct an informative Armstrong ABox for uniqueness- and null-free constraints for an n-ary relation C, we need to know about all uniqueness- and null-free constraints that apply to C. For this reason we propose using data mining routines to determine the uniqueness- and null-free constraints realized by the actual data for an n-ary relation C, which can then be used to generate a related informative Armstrong ABox.

Determining constraints from an ABox is problematic due to the open world assumption of DLs – the absence of information does not imply non-existence of information as is the case with the closed world assumption. We deal with this challenge in a similar fashion as Patel-Schneider [9]. We assume that an ABox completely describes a world and therefore make the following assumptions: (1) we assume an assertion is false if it does not follow from an ontology, which is referred to as the close world assumption, and (2) individuals with different names represent different individuals, which is called the unique name assumption.

This section is structured as follows: In Section 3.1 we define Armstrong ABoxes for uniqueness- and null-free constraints of n-ary relations with the goal to be usable in the context of the Semantic Web. Section 3.2 illustrates how data mining routines can be used to determine n-ary relations, uniqueness- and null-free constraints for a given ABox. Section 3.3 explains the algorithm for generating an informative Armstrong ABox. In Section 3.4 we apply the notion of an informative Armstrong ABox to a DBPedia person dataset to derive an informative Armstrong RDF dataset. Section 3.5 summarizes the potential benefits of Armstrong RDF datasets.

3.1. Armstrong ABoxes for n-ary Relations

Armstrong ABoxes have been introduced as a DL counterpart to the Armstrong relations of relational database theory, in order to support ontology engineers in designing ontologies [4]. The Armstrong ABox formalization of Harmse et al. adheres strictly to the relational database counterpart. In order to introduce the notion of an Armstrong RDF dataset we need to relax the definition slightly, which we illustrate via an example.

Assume we need to model the schedule at a university where a course is presented by a lecturer at a given time in a specific room. This can be modelled in OWL 2 using reification as follows:

$$\mathcal{T}_{\mathsf{Schedule}} = \{$$

$$\mathsf{Schedule} \sqsubseteq \exists \mathsf{course}.\top \sqcup \exists \mathsf{time}.\top \sqcup \exists \mathsf{lecturer}.\top \sqcup \exists \mathsf{room}.\top, \tag{1}$$

$$\top \sqsubseteq \forall \mathsf{course}.\mathsf{Course}, \top \sqsubseteq \forall \mathsf{time}.\mathsf{Time}, \top \sqsubseteq \forall \mathsf{lecturer}.\mathsf{Lecturer}, \top \sqsubseteq \forall \mathsf{room}.\mathsf{Room}, \tag{2}$$

$$\mathsf{Schedule}\ \mathtt{hasKey}\ (\mathsf{time}), \mathsf{Schedule}\ \mathtt{hasKey}\ (\mathsf{course}, \mathsf{lecturer}), \tag{3}$$

$$\mathsf{Schedule}\ \mathtt{hasKey}\ (\mathsf{lecturer}, \mathsf{room}), \tag{4}$$

$$\mathsf{Schedule} \sqsubseteq \exists \mathsf{course}.\top, \mathsf{Schedule} \sqsubseteq \exists \mathsf{time}.\top\}. \tag{5}$$

(1) states that individuals of $\mathsf{Schedule}$ represents an n-ary relation that can have up to 4 components, namely course, time, $\mathsf{lecturer}$, $\mathsf{room} \in N_R$, such that each individual of $\mathsf{Schedule}$ can be associated zero or multiple times with each component. Note that due to the disjunction this includes individuals of $\mathsf{Schedule}$ having 0, 1, 2 or 3 components from $\{\mathsf{course}, \mathsf{time}, \mathsf{lecturer}, \mathsf{room}\}$. Moreover, for our initial research into Armstrong RDF datasets we decided to define an n-ary relation, as for example for $\mathsf{Schedule}$, as the maximal set S such that each $r \in S$ has $\mathsf{Schedule}$ as a domain. In future research it may make sense to discover minimal set covers of S which can be used to decompose an n-ary relation like $\mathsf{Schedule}$.

(2) enforces the typing of the components of $\mathsf{Schedule}$. I.e. the course component must be of type Course where $\mathsf{Course} \in \triangle^{\mathcal{I}} \cup \triangle^{\mathcal{D}}$. Hence, (1) and (2) state that the **domain** and **range** of course is respectively $\mathsf{Schedule}$ and Course. (3) and (4) enforces the set of **uniqueness constraints** $\Sigma = \{\mathsf{u}(\mathsf{time}), \mathsf{u}(\mathsf{course}, \mathsf{lecturer}), \mathsf{u}(\mathsf{lecturer}, \mathsf{room})\}$ on $\mathsf{Schedule}$. For example, based on the uniqueness constraint $\mathsf{u}(\mathsf{time})$, no two different individuals of $\mathsf{Schedule}$ agree on their participation to the component time. (5) states that $\{\mathsf{course}, \mathsf{time}\}$ is **null-free** on $\mathsf{Schedule}$: every individual c of $\mathsf{Schedule}$ is associated with individuals via roles course and time.

Now assume we have an ABox $\mathcal{A}_{\mathsf{Schedule}}$ as defined in Table 1. Note that we expect explicit assertions stating that, for example, individuals c_1, c_2, c_3 and c_4 are of type $\mathsf{Schedule}$. Then we say c_2 is $\{\mathsf{course}, \mathsf{time}, \mathsf{lecturer}, \mathsf{room}\}$**-total**, since values are specified for these components for individual c_2. We say $\mathcal{A}_{\mathsf{Schedule}}$ is $\{\mathsf{course},$

Table 1. ABox $\mathcal{A}_{\mathsf{Schedule}}$ specifying facts regarding $\mathsf{Schedule}$

Concept assertions			
$\mathsf{Schedule}(c_1)$	$\mathsf{Schedule}(c_2)$	$\mathsf{Schedule}(c_3)$	$\mathsf{Schedule}(c_4)$
Role assertions			
$\mathsf{course}(c_1, Math11)$	$\mathsf{time}(c_1, \text{“}Mon, 2pm\text{”})$	$\mathsf{lecturer}(c_1, Russel)$	$\mathsf{room}(c_1, Red)$
$\mathsf{course}(c_2, Math11)$	$\mathsf{time}(c_2, \text{“}Tue, 2pm\text{”})$	$\mathsf{lecturer}(c_2, Hardy)$	$\mathsf{room}(c_2, Red)$
$\mathsf{course}(c_3, Math11)$	$\mathsf{time}(c_3, \text{“}Wed, 1pm\text{”})$		$\mathsf{room}(c_3, Red)$
$\mathsf{course}(c_6, Math12)$	$\mathsf{time}(c_6, \text{“}Wed, 4pm\text{”})$	$\mathsf{lecturer}(c_6, Hardy)$	$\mathsf{room}(c_6, Black)$
$\mathsf{course}(c_9, Math21)$	$\mathsf{time}(c_9, \text{“}Fri, 1pm\text{”})$	$\mathsf{lecturer}(c_9, Peter)$	

$\mathsf{time}\}$-total since all named individuals of $\mathsf{Schedule}$ have values for components course and time, in which case we also say $\mathcal{A}_{\mathsf{Schedule}}$ is **null-free** for $\{\mathsf{course}, \mathsf{time}\}$. The **strong agreement set** of individuals c_2 and c_3, denoted by $\mathsf{ag}^s(c_2, c_3)$, is $\{\mathsf{course}, \mathsf{room}\}$ since individuals c_2 and c_3 agree on their participation for the components course, and room. The **strong agreement set** of $\mathcal{A}_{\mathsf{Schedule}}$ is given by

$$\begin{aligned} \mathsf{ag}^s(\mathcal{A}_{\mathsf{Schedule}}) &= \{\mathsf{ag}^s(c_i, c_j) | \mathsf{Schedule}(c_i) \wedge \mathsf{Schedule}(c_j) \wedge c_i \not\approx c_j\} \\ &= \{\{\mathsf{course}, \mathsf{room}\}, \{\mathsf{course}\}, \{\mathsf{lecturer}\}, \{\mathsf{room}\}\} \end{aligned} \tag{6}$$

The ABox $\mathcal{A}_{\mathsf{Schedule}}$ of Table 1 is an **Armstrong ABox** for $\mathcal{T}_{\mathsf{Schedule}}$ since it satisfies all uniqueness- and null-free constraints defined by $\mathcal{T}_{\mathsf{Schedule}}$ and it also violates all other uniqueness- and null-free constraints that cannot be derived from $\mathcal{T}_{\mathsf{Schedule}}$. We say that $\mathcal{A}_{\mathsf{Schedule}}$ **satisfies** $\Sigma = \{\mathsf{u}(\mathsf{time}),\ \mathsf{u}(\mathsf{course}, \mathsf{lecturer}),\ \mathsf{u}(\mathsf{lecturer}, \mathsf{room})\}$. However, for $\Sigma = \{\mathsf{u}(\mathsf{course}), \mathsf{u}(\mathsf{room})\}$ we say that $\mathcal{A}_{\mathsf{Schedule}}$ **violates** Σ. Note that even though the ABox of Table 1 is indeed an Armstrong ABox for the applicable uniqueness- and null-free constraints, it is not necessarily a minimal Armstrong ABox. For example, if we remove the assertions of the first row from table Table 1 from $\mathcal{A}_{\mathsf{Schedule}}$, what is left will still be an Armstrong ABox for $\mathcal{T}_{\mathsf{Schedule}}$.

3.2. Data Mining Routines for ABoxes

In this section we present the data mining algorithm for determining whether a candidate n-ary relation C represents an n-ary relation, and if it does, the algorithm will determine the uniqueness constraints and null-free constraints applicable to the n-ary relation C. For this purpose we introduce DETERMINESTRUCTURE($\mathcal{T}$, $\mathcal{A}$, C) (see Algorithm 1) which has as input parameters, a TBox $\mathcal{T}$, an ABox $\mathcal{A}$ and a concept C, which is a candidate n-ary relation. On termination

- DETERMINESTRUCTURE returns the empty set or a singleton set if C does not represent an n-ary relation, or
- if C does represent an n-ary relation, it returns a set S of roles that serve as the components of C, a set X which is a subset of S representing the roles that are null-free for C, and a set Σ consisting of uniqueness constraints applicable to C.

The key steps of the algorithm are:

Step 1 - Determine S: In line 2 we determine the set of roles S for which C is the domain, which represents the components of the n-ary relation represented by C. If the set S is a singleton set or the empty set, we assume that C does not represent an n-ary relation and the algorithm terminates.

Step 2 - Determine X: Line 4 determines the set of individuals that are elements of C. For each individual $c \in T$ we check via which roles in S is c related to another individual. When c is not related to an individual for some role $r \in S$, we know that r is a component of the n-ary relation represented by C that is nullable. Therefore we add r to H in line 8. The roles that are null-free for C can be determined from H and S (line 9).

Step 3 - Determine Σ: For calculating the uniqueness constraints satisfied by an ABox $\mathcal{A}$, we use ideas inspired by Le et al. [7]. Line 10 computes the complements of strong agreement sets called **weak agreement sets**, which is denoted by $\mathsf{disag}^w(\mathcal{A})$ for an ABox $\mathcal{A}$. Line 11 computes the **necessary disagreement sets** that is the weak agreement sets that are minimal, which is denoted by $\mathsf{nec\_disag}^w(\mathcal{A})$ for an ABox $\mathcal{A}$. Line 12 constructs a hypergraph $\mathcal{H}$ with S as vertices and $\mathsf{nec\_disag}^w(\mathcal{A})$ as edges. Lemma 3.1 confirms that calculating the minimal transversal of $\mathcal{H}$, denoted by $\mathsf{Tr}(\mathcal{H})$, will deliver the set of minimal uniqueness constraints satisfied by the ABox $\mathcal{A}$.

Lemma 3.1. *Let $\mathcal{A}$ be an ABox and C a concept representing an n-ary relation over the set of roles $S = \{r_1, \ldots, r_n\}$. Then for all $X \subseteq S$, $X \in \mathsf{Tr}(S, \mathsf{nec\_disag}^w(\mathcal{A}))$ if and only if $\mathcal{A} \Vdash \mathsf{u}(X)$ and for all $H \in X$, $\mathcal{A} \nVdash \mathsf{u}(X \backslash H)$.* ◊

Algorithm 1 DetermineStructure($\mathcal{T}$, $\mathcal{A}$, C)

```
1: $\Sigma := \varnothing$, $H := \varnothing$, $\mathcal{O} := (\mathcal{T}, \mathcal{A})$
                                                        ▷ Determine $S$
2: $S := \{\mathbf{r} | \text{There exists some } \mathbf{c}, \mathbf{h} \text{ s.t. } \mathbf{r}(\mathbf{c}, \mathbf{h}) \in P\}$ where $P := \mathsf{ans}(\mathcal{O}, \{C(\mathbf{c}), \mathbf{r}(\mathbf{c}, \mathbf{h})\})$
3: if $\sharp S \leq 1$ then return $S$
                                                        ▷ Determine $X$
4: $T := \mathsf{ans}(\mathcal{O}, \{C(\mathbf{c})\})$
5: for all $c \in T$ do
6:     for all $r \in S$ do
7:         if $\mathsf{ans}(\mathcal{O}, \{r(c, \mathbf{h})\}) = \varnothing$ then
8:             $H := H \cup \{r\}$
9: $X := S \backslash H$
                                                        ▷ Determine $\Sigma$
10: $\mathsf{disag}^w(\mathcal{A}) := \{S \backslash \mathsf{ag}^s(c_1, c_2) | C(c_1) \wedge C(c_2) \wedge c_1 \not\approx c_2\}$
11: $\mathsf{nec\_disag}^w(\mathcal{A}) := \{W \in \mathsf{disag}^w(\mathcal{A}) | \neg \exists Y \in \mathsf{disag}^w(\mathcal{A})(Y \subset W)\}$
12: $\mathcal{H} := (S, \mathsf{nec\_disag}^w(\mathcal{A}))$
13: $\Sigma := \{\mathsf{u}(Y) | Y \in \mathsf{Tr}(\mathcal{H})\}$
14: return $S$, $X$, $\Sigma$
```

Algorithm 2 InformativeArmstrongABox($\mathcal{T}$, $\mathcal{A}$, C, S, Σ, X)

```
1: $\mathcal{A}_{\mho} := \varnothing$
                                                    ▷ Determine first tuple
2: $i := 0$
3: $\mathcal{Q} := \{C(\mathbf{c})\}$
4: for all $r_i \in S$ do
5:     $\mathcal{Q} := \mathcal{Q} \cup \{r_i(\mathbf{c}, \mathbf{h}_i)\}$
6:     $i := i + 1$
7: ConditionalAdd1($\mathcal{A}$, $\mathcal{A}_{\mho}$, $\mathcal{Q}$, $S$)
                                                    ▷ Deal with $\Sigma^{-1}$
8: $\mathcal{H} := (S, Z)$ where $\{Z | \mathsf{u}(Z) \in \Sigma\}$
9: $\Sigma^{-1} := \{\mathsf{a}(S \backslash W) | W \in \mathsf{Tr}(\mathcal{H})\}$
10: for all $Y$ such that $a(Y) \in \Sigma^{-1}$ do
11:     $\mathcal{Q} := \{C(\mathbf{c_1}), C(\mathbf{c_2})\}$
12:     $i := 0$
13:     for all $r_i \in S$ do
14:         $\mathcal{Q} := \mathcal{Q} \cup \begin{cases} \{r_i(\mathbf{c_1}, \mathbf{h}_i), r_i(\mathbf{c_2}, \mathbf{h}_i)\}, \text{ if } r_i \in Y \\ \{r_i(\mathbf{c_1}, \mathbf{h}_{i_1}), r_i(\mathbf{c_2}, \mathbf{h}_{i_2})\}, \text{ else} \end{cases}$;
15:         $i := i + 1$
16:     ConditionalAdd2($\mathcal{T}$, $\mathcal{A}$, $\mathcal{A}_{\mho}$, $\mathcal{Q}$, $S$)
                                                    ▷ Deal with $S \backslash X$
17: Let $Z \subseteq S$ be such that $\mathcal{A}_{\mho}$ is $Z$-total
18: if $Z \backslash X \neq \varnothing$ then
19:     for all $r \in Z \backslash X$ do
20:         $\mathcal{Q} := \{C(\mathbf{c}), (\neg \exists r.\top)(\mathbf{c})\}$
21:         ConditionalAdd1($\mathcal{A}$, $\mathcal{A}_{\mho}$, $\mathcal{Q}$, $S$)
22:     return $\mathcal{A}_{\mho}$
23: else
24:     return $\mathcal{A}_{\mho}$
```

3.3. Steps for Generating an Informative Armstrong ABox

InformativeArmstrongABox($\mathcal{T}$, $\mathcal{A}$, C, S, Σ, X) (Algorithm 2) generates an informative Armstrong ABox $\mathcal{A}_{\mho}$, where $\mathcal{T}$ is a TBox, $\mathcal{A}$ is the actual ABox containing assertions about the real-world, C is a concept representing an n-ary relation, S represents the components of the relation C, Σ is the set of uniqueness constraints and X is the set of roles that are null-free. We assume $\mathcal{O} := (\mathcal{T}, \mathcal{A})$ and $\mathcal{O}_{\mho} := (\mathcal{T}, \mathcal{A}_{\mho})$.

The algorithm consists of three main steps. For steps 2 and 3 it first does a query against $\mathcal{O}_{\mho}$ to determine whether the required assertions are already present in $\mathcal{A}_{\mho}$. If the answer already follows from $\mathcal{O}_{\mho}$, there is no need to add assertions to $\mathcal{A}_{\mho}$. However, if the answer does not follow from $\mathcal{O}_{\mho}$, a query is done against $\mathcal{O}$ and the answer can be used to add the required assertions to $\mathcal{A}_{\mho}$. Since the n-ary relation and uniqueness- and null-free constraints were determined using data mining routines against $\mathcal{O} := (\mathcal{T}, \mathcal{A})$, we know that all queries against $\mathcal{O}$ (based on this n-ary relation, uniqueness- and null-free constraints) will succeed.

We first describe the main steps of the InformativeArmstrongABox algorithm and then we describe the auxiliary algorithms ConditionalAdd1, ConditionalAdd2 and AddRoleFiller.

Step 1 - First tuple: For our first tuple we ensure that the individual c of C is such that for each $r_i \in S$ we have that c has an associated value for r_i (line 2-7).

Step 2 - Deal with Σ^{-1}: The set of anti-keys, Σ^{-1}, is calculated using hypergraph methods. Line 8 constructs the hypergraph $\mathcal{H} := (S, Z)$ where Z consists of sets of components that are unique for C. In line 9 Σ^{-1} is calculated based on the minimal transversal of $\mathcal{H}$, denoted by $\mathsf{Tr}(\mathcal{H})$ (see Lemma 1 of Harmse et al [4]). For each anti-key $\mathsf{a}(Y)$ in Σ^{-1}, a query $\mathcal{Q}$ is constructed to find two instances c_1 and c_2 of C such that they agree on their participation to the roles in Y (lines 10-16).

Step 3 - Deal with $S\backslash X$: Algorithm 2, lines 17-24 ensure that for each $r \notin X$ there is at least one individual c of C such that c that does not have a role filler for r (line 20). To determine whether there are any $r \in S\backslash X$ for which we need to add assertions, it gets the difference between Z and X (line 18), where Z consists of the roles for which $\mathcal{A}_{\sqcup}$ is null-free this far. This completes the algorithm for generating an informative Armstrong ABox $\mathcal{A}_{\sqcup}$.

Algorithm 3 CONDITIONALADD1($\mathcal{T}$, $\mathcal{A}$, $\mathcal{A}_{\sqcup}$, $\mathcal{Q}$, S)

```
Ensure: $\mathcal{Q}$ is of the form $\{C(\mathbf{c}), r_1(\mathbf{c},\mathbf{h_1}),\dots,r_k(\mathbf{c},\mathbf{h_k})\}$ where $\{r_1,\dots,r_k\} \subseteq S$
1: $\mathcal{O} := (\mathcal{T},\mathcal{A})$, $\mathcal{O}_{\sqcup} := (\mathcal{T},\mathcal{A}_{\sqcup})$
2: if $\mathsf{ans}(\mathcal{O}_{\sqcup},\mathcal{Q}) \neq \varnothing$ then return $\mathcal{A}_{\sqcup}$
3: $\mu(\mathcal{Q})$ such that $\mathcal{O} \models \mu(\mathcal{Q})$
4: $\mathcal{A}_{\sqcup} := \mathcal{A}_{\sqcup} \cup \{C(\mu(\mathbf{c})), r_1(\mu(\mathbf{c}),\mu(\mathbf{h_1})),\dots,r_k(\mu(\mathbf{c}),\mu(\mathbf{h_k}))\}$
5: $Y := S - \{r_1,\dots,r_k\}$
6: if $Y = \varnothing$ then return $\mathcal{A}_{\sqcup}$
7: for all $r \in Y$ do
8:     ADDROLEFILLER($\mathcal{T}$, $\mathcal{A}$, $\mathcal{A}_{\sqcup}$, $\{C(\mathbf{c}), r(\mathbf{c},\mathbf{h})\}$)
9: return $\mathcal{A}_{\sqcup}$
```

Algorithm 4 CONDITIONALADD2($\mathcal{T}$, $\mathcal{A}$, $\mathcal{A}_{\sqcup}$, $\mathcal{Q}$, S)

```
Ensure: $\mathcal{Q}$ is of the form $\{C(\mathbf{c_1}),C(\mathbf{c_2}),r_1(\mathbf{c_1},\mathbf{h_{1_1}}),\dots,r_k(\mathbf{c_1},\mathbf{h_{k_1}}),r_1(\mathbf{c_2},\mathbf{h_{1_2}}),\dots,$
    $r_k(\mathbf{c_2},\mathbf{h_{k_2}})\}$ where $\{r_1,\dots,r_k\} \subseteq S$
1: $\mathcal{O} := (\mathcal{T},\mathcal{A})$, $\mathcal{O}_{\sqcup} := (\mathcal{T},\mathcal{A}_{\sqcup})$
2: if $\mathsf{ans}(\mathcal{O}_{\sqcup},\mathcal{Q}) \neq \varnothing$ then return $\mathcal{A}_{\sqcup}$
3: $\mu(\mathcal{Q})$ such that $\mathcal{O} \models \mu(\mathcal{Q})$
4: $\mathcal{A}_{\sqcup} := \mathcal{A}_{\sqcup} \cup \{C(\mu(\mathbf{c_1})), C(\mu(\mathbf{c_2})), r_1(\mu(\mathbf{c_1}),\mu(\mathbf{h_{1_1}})),\dots,r_k(\mu(\mathbf{c_1}),\mu(\mathbf{h_{k_1}})),$
              $r_1(\mu(\mathbf{c_2}),\mu(\mathbf{h_{1_2}})),\dots,r_k(\mu(\mathbf{c_2}),\mu(\mathbf{h_{k_2}}))\}$
5: $Y := S\backslash\{r_1,\dots,r_k\}$
6: if $Y = \varnothing$ then return $\mathcal{A}_{\sqcup}$
7: for all $r \in Y$ do
8:     ADDROLEFILLER($\mathcal{T}$, $\mathcal{A}$, $\mathcal{A}_{\sqcup}$, $\{C(\mathbf{c_1}), r(\mathbf{c_1},\mathbf{h})\}$)
9:     ADDROLEFILLER($\mathcal{T}$, $\mathcal{A}$, $\mathcal{A}_{\sqcup}$, $\{C(\mathbf{c_2}), r(\mathbf{c_2},\mathbf{h})\}$)
10: return $\mathcal{A}_{\sqcup}$
```

Both the CONDITIONALADD1($\mathcal{T}$, $\mathcal{A}$, $\mathcal{A}_{\sqcup}$, $\mathcal{Q}$, S) and CONDITIONALADD2($\mathcal{T}$, $\mathcal{A}$, $\mathcal{A}_{\sqcup}$, $\mathcal{Q}$, S) algorithms are used to add assertions to $\mathcal{A}_{\sqcup}$ conditionally. The main difference between these two algorithms is that CONDITIONALADD1 adds assertions representing one individual and CONDITIONALADD2 adds assertions representing two individuals of the n-ary relation C to $\mathcal{A}_{\square}$. Since these two algorithms are very similar, we will only discuss CONDITIONALADD2.

CONDITIONALADD2($\mathcal{A}$, $\mathcal{A}_{\square}$, $\mathcal{Q}$, S) (Algorithm 4) adds, if needed, assertions corresponding to the answer $\mu(\mathcal{Q})$ to $\mathcal{A}_{\square}$. We say "if needed" because it is possible that $\mathcal{Q}$ can already be answered from $\mathcal{O}_{\square}$, in which case it is not necessary to add assertions corresponding to $\mu(\mathcal{Q})$ to $\mathcal{A}_{\square}$ and Algorithm 4 can return (line 2).

CONDITIONALADD2 enforces that query $\mathcal{Q}$ is of a specific form (see `Ensure` statement): it consists of two individuals c_1 and c_2 of concept C that agrees on their participation to roles $\{r_1, \ldots, r_k\} \subseteq S$. Since $\{r_1, \ldots, r_k\}$ is potentially a subset of S, it is possible that an answer does not have all components of C represented. This is the reason for lines 5-9. We determine the difference Y between S and $\{r_1, \ldots, r_k\}$ and if Y is not the empty set, for each r in Y an attempt is made to fill r for individuals c_1 and c_2 by calling the ADDROLEFILLER(Algorithm 5).

ADDROLEFILLER enforces that $\mathcal{Q}$ is of the form $\{C(\mathbf{c}), r(\mathbf{c}, \mathbf{h})\}$. If an answer $\mu(\mathcal{Q})$ exists such that $\mathcal{O} \vDash \mu(\mathcal{Q})$, corresponding assertions are added to $\mathcal{A}_{\square}$, otherwise the procedure returns.

Algorithm 5 ADDROLEFILLER($\mathcal{T}$, $\mathcal{A}$, $\mathcal{A}_{\square}$, $\mathcal{Q}$)

```
Ensure: $\mathcal{Q}$ is of the form $\{C(\mathbf{c}), r(\mathbf{c}, \mathbf{h})\}$
1: $\mathcal{O} := (\mathcal{T}, \mathcal{A})$
2: if $\mathsf{ans}(\mathcal{O}, \{C(\mathbf{c}), r(\mathbf{c}, \mathbf{h})\}) = \varnothing$ then return $\mathcal{A}_{\square}$
3: $\mathcal{A}_{\square} := \mathcal{A}_{\square} \cup \{r(\mu(\mathbf{c}), \mu(\mathbf{h}))\}$
4: return $\mathcal{A}_{\square}$
```

3.4. A Case Study

In this case study we applied the proposed approach in this paper on the English person data set of 2016-10 from DBPedia [1]. The code implementing the algorithms in this paper along with the resulting informative Armstrong RDF dataset is available publicly from a `git` repository [1]. Here we highlight some key results from our case study.

We imported the DBPedia dataset into a local Ontotext GraphDB Free Edition RDF store. We implemented our data mining algorithm and informative Armstrong generation algorithms in Java 8. Test runs were done with GraphDB and the data mining and Armstrong routines co-located on a laptop with an i7-2760QM CPU at 2.40GH and 8GB of RAM running on a Ubuntu 17.10 desktop operating system.

[1]`https://github.com/henrietteharmse/ArmstrongDBPediaCaseStudy`

For the case study we assumed C = `http://dbpedia.org/ontology/Person`. Using Algorithm 1 we found that $X = \varnothing$ and S =

{`http://xmlns.com/foaf/0.1/name`,
`http://xmlns.com/foaf/0.1/surname`,
`http://xmlns.com/foaf/0.1/givenName`,
`http://xmlns.com/foaf/0.1/gender`,
`http://purl.org/dc/terms/description`,
`http://dbpedia.org/ontology/birthDate`,
`http://dbpedia.org/ontology/birthPlace`,
`http://dbpedia.org/ontology/deathDate`,
`http://dbpedia.org/ontology/deathPlace`}.

Σ, the set of uniqueness constraints for C and S is given in Table 2. Recall that Σ is a set of subsets of S. Each row of Table 2 represents a subset of S, where a cross ($\times$) indicates that an element is part of the subset, and absence of a cross indicates that the element is not included in the subset. As an example, the first row of the left-hand side column of Table 2 states that `u(name, surname, givenName, gender, deathPlace)` is a uniqueness constraint of C = `http://dbpedia.org/ontology/Person`.

Based on S, X and Σ we determined an informative Armstrong RDF dataset for C using Algorithm 2. In Table 3 we compare some statistics for the DBPedia dataset and the related informative Armstrong RDF dataset. From the number of person instances and statements regarding persons, it is evident that the size of the informative Armstrong RDF dataset is about 0.00003% of the DBPedia dataset. This reduction in size of the dataset can be of benefit to applications. In our situation some of the queries for determining the strong agreement set for C and S ran just over 30 minutes. The same query took less than 1 second to run on the Armstrong dataset. Running the data mining routines and generating the Armstrong dataset based on the DBPedia dataset took about 5.7 hours. To validate our implementation of the data mining and Armstrong dataset generation routines, we applied the data mining and Armstrong generation routines to the generated Armstrong dataset. As expected the newly calculated sets S, X and Σ matched our earlier calculations, but it only took 8 seconds to run both data mining and Armstrong generation routines.

Table 2. Uniqueness constraints for the person data of DBPedia

`name`	`surname`	`givenName`	`gender`	`description`	`birthDate`	`birthPlace`	`deathDate`	`deathPlace`	`name`	`surname`	`givenName`	`gender`	`description`	`birthDate`	`birthPlace`	`deathDate`	`deathPlace`
×	×	×	×	×				×			×	×		×	×	×	
	×		×	×	×	×		×				×	×	×	×	×	×
×	×	×	×		×		×				×	×		×		×	×
×	×	×	×			×	×	×			×	×	×		×	×	×
	×		×	×		×	×	×		×		×		×	×	×	
	×		×	×	×		×			×	×	×	×		×		×
×			×		×			×		×	×	×	×	×	×		
		×	×	×	×	×		×	×	×	×	×	×		×		
		×	×	×	×		×		×	×	×	×		×	×		

Table 3. Informative Armstrong RDF dataset characteristics

Characteristic	**DBPedia dataset**	**Armstrong dataset**
Person instances	1 414 214	44
Explicit person statements	10 252 799	373
Worst query running time	30 minutes	< 1 second
Total running time	5.7 hours	8 seconds

3.5. The Potential Benefits of an Informative Armstrong RDF Dataset

Generating an informative Armstrong RDF dataset has a number of potential benefits:

1. the set S of components identified for a candidate n-ary relation C, along with uniqueness- and null-free constraints can be used to construct an ontology for the data similar to axioms (1)-(5),
2. users can use the uniqueness constraints to determine how to identify individuals uniquely,
3. based on the null-free constraints users can determine which components of C are nullable, which would guide them on when to specify the SPARQL `OPTIONAL` keyword when constructing queries,
4. based on the uniqueness constraints RDF storage implementers can fine-tune indexing accordingly, and
5. an informative Armstrong RDF dataset give users and Semantic Web application developers the opportunity to verify queries and algorithms against a small representative example dataset, which is likely to give comparatively quick feedback, before running their queries and/or applications against the actual dataset.

4. Conclusion

In this paper we proposed an approach of how data mining routines can be used to construct an informative Armstrong RDF dataset based on an informative Armstrong ABox for n-ary relations with uniqueness- and null-free constraints, which can assist users in understanding the structure of data captured through a data-first approach. The benefit of an informative Armstrong ABox is that it provides example data based on the original data which is a fraction of the size of the original data, while capturing the constraints of the original data faithfully. Working on a fraction of the data has the potential to increase developer productivity substantially and understanding the structure of RDF data gives insight into how to query the data and how to query the data efficiently.

To validate our algorithms we have generated an informative Armstrong RDF dataset for the DBPedia person dataset. The results from the case study indicate that an informative Armstrong RDF dataset can result in a dataset that contains 0.00003% of the statements in the original dataset. Further research is required to determine whether this result can be extended to RDF datasets in general. Nonetheless, this result seems promising and merits further investigation.

References

[1] *DBPedia Person Data 2016 - English*, http://downloads.dbpedia.org/2016-10/core-i18n/en/persondata_en.ttl.bz2.

[2] Ronald Fagin and Moshe Y. Vardi, *Armstrong Databases for Functional and Inclusion Dependencies.*, Inf. Process. Lett. **16** (1983), no. 1, 13–19.

[3] B. Glimm, Y. Kazakov, I. Kollia, and G. B. Stamou, *Using the TBox to Optimise SPARQL Queries.*, Description Logics (T. Eiter, B. Glimm, Y. Kazakov, and M. Krötzsch, eds.), CEUR Workshop Proceedings, vol. 1014, CEUR-WS.org, 2013, pp. 181–196.

[4] H. F. Harmse, A. Britz, and A. Gerber, *Armstrong Relations for Ontology Design and Evaluation*, Proceedings of the 29th International Workshop on Description Logics (M. Lenzerini and R. Peñaloza, eds.), vol. 1577, CEUR-WS.org, 2016.

[5] A. Hogan, *Linked Data & the Semantic Web Standards.*, Linked Data Management (A. Harth, K. Hose, and R. Schenkel, eds.), Chapman and Hall/CRC, 2014, pp. 3–48.

[6] I. Horrocks, O. Kutz, and U. Sattler, *The even more irresistible $\mathcal{SROIQ}$*, Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (P. Doherty, J. Mylopoulos, and C. A. Welty, eds.), AAAI Press, 2006, pp. 57–67.

[7] V. B. T. Le, S. Link, and M. Memari, *Schema- and Data-driven Discovery of SQL Keys.*, Journal of Computing Science and Engineering **6** (2012), no. 3, 193–206.

[8] B. Parsia, U. Sattler, and T. Schneider, *Easy Keys for OWL*, Proceedings of the 5th Workshop on OWL: Experiences and Directions (C. Dolbear, A. Ruttenberg, and U. Sattler, eds.), CEUR Workshop Proceedings, vol. 432, CEUR Workshop Proceedings, 2008.

[9] P. F. Patel-Schneider, *Using Description Logics for RDF Constraint Checking and Closed-World Recognition.*, Proceedings of the 29th AAAI Conference on Artificial Intelligence, AAAI Press, 2015.

[10] M. Pham, L. Passing, O. Erling, and P. A. Boncz, *Deriving an Emergent Relational Schema from RDF Data.*, 24th International World Wide Web Conference (A. Gangemi, S. Leonardi, and A. Panconesi, eds.), ACM, May 2015, pp. 864–874.