Link key candidate extraction with relational concept analysis

Manuel Atencia, Jérôme David, Jérôme Euzenat, Amedeo Napoli, Jérémy Vizzini

Link key candidate extraction with relational concept analysis

Manuel Atencia[a], Jérôme David[a], Jérôme Euzenat[a,*], Amedeo Napoli[b], Jérémy Vizzini[a]

[a] *Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France*
[b] *Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France*

Abstract

Linked data aims at publishing data expressed in RDF (Resource Description Framework) at the scale of the worldwide web. These datasets interoperate by publishing links which identify individuals across heterogeneous datasets. Such links may be found by using a generalisation of keys in databases, called link keys, which apply across datasets. They specify the pairs of properties to compare for linking individuals belonging to different classes of the datasets. Here, we show how to recast the proposed link key extraction techniques for RDF datasets in the framework of formal concept analysis. We define a formal context, where objects are pairs of resources and attributes are pairs of properties, and show that formal concepts correspond to link key candidates. We extend this characterisation to the full RDF model including non functional properties and interdependent link keys. We show how to use relational concept analysis for dealing with cyclic dependencies across classes and hence link keys. Finally, we discuss an implementation of this framework.

Keywords: Formal Concept Analysis, Relational Concept Analysis, Linked data, Link key, Data interlinking, Resource Description Framework

1. Motivations

The linked data initiative aims at exposing, sharing and connecting structured data on the web [10, 24]. Linked data includes a large amount of data in the form of RDF (Resource Description Framework [16]) triples, described using terms of RDF Schema and OWL ontologies [33]. It has published more than 149 billions of triples distributed over datasets from many different domains[1] (media, life sciences, geography). This has attracted many organisations that today publish and consume linked data, ranging from private companies like the BBC or the New York Times, to public institutions like Ordnance Survey or the British Library in the UK or Insee and BNF (Bibliothèque Nationale de France) in France.

An important added value of linked data arises from the "same-as" links, which identify the same entity in different datasets. Innovative applications exploit such cross-references and make inferences across datasets. Therefore, the task of deciding whether two resources described in RDF over possibly different datasets refer to the same real-world entity is critical for widening and enhancing linked data. This task, that we refer to as data interlinking, is a knowledge discovery task as it infers knowledge —the condition for identity of objects— from data.

Different approaches and methods have been proposed to address the problem of automatic data interlinking [21, 34]. Most of them are based on numerical methods that measure a similarity between entities and consider that the closest the entities, the more likely they are the same. These methods typically output weighted links, with the links assigned the higher weights expected to be correct [49, 35]. A few other works take a logical approach to data interlinking and can leverage reasoning methods [42, 3, 2, 25]. One of these approaches relies on *link keys* [5].

Link keys generalise keys from relational databases to different RDF datasets. An example of a link key is:

$$\{\langle \mathsf{auteur}, \mathsf{creator}\rangle\}\{\langle \mathsf{titre}, \mathsf{title}\rangle\} \; \mathit{linkkey} \; \langle \mathsf{Livre}, \mathsf{Book}\rangle$$

*Corresponding author
Email addresses: Manuel.Atencia@inria.fr (Manuel Atencia), Jerome.David@inria.fr (Jérôme David), Jerome.Euzenat@inria.fr (Jérôme Euzenat), Amedeo.Napoli@loria.fr (Amedeo Napoli), jeremyvizzini@icloud.com (Jérémy Vizzini)

[1] As reported for the CKAN data hub by http://stats.lod2.eu/ (2017-02-20) which can now be consulted through internet archive https://web.archive.org/web/20170220115805/http://stats.lod2.eu/

stating that whenever an instance of the class Livre has the same values for the property auteur as an instance of the class Book has for the property creator and they share at least one value for their properties titre and title, then they denote the same entity. The notion of link key used in this paper generalises the definition introduced in [5] because the body of the rule includes two sets: the set of property pairs for which instances have to share all the values and the set of property pairs for which instances have to share at least one value.

Such a link key may depend on other ones. For instance, properties auteur and creator may have values in the Écrivain and Writer classes respectively. Identifying their values will then resort to another link key:

$$\{\langle \text{prénom}, \text{firstname}\rangle\}\{\langle \text{nom}, \text{lastname}\rangle\}\ \textit{linkkey}\ \langle \text{Écrivain}, \text{Writer}\rangle$$

This situation would be even more intricate if Écrivain and Writer were instead identified from the values of their properties ouvrages and hasWritten refering to instances of Livre and Book (which have a chance to be more accurate). We would then face interdependent link keys.

The problem considered here is the extraction of such link keys from RDF data. We have already proposed an algorithm for extracting some types of link keys [5]. This method may be decomposed in two distinct steps: (1) identifying link key candidates, i.e. sets of property pairs that would generate at least one link if used as a link key and that would be maximal for at least one generated link, followed by (2) selecting the best link key candidates according to quality measures. A method for discovering functional link keys in relational databases based on Formal Concept Analysis (FCA [23]) is detailed in [6].

Globally extracting a set of link keys across several RDF data sources raises several issues, as link keys differ from database keys in various aspects: (*a*) they relax two constraints of the relational model, namely, that attributes are functional (RDF properties may have several values) and that attribute values are data types (RDF property values may be objects too) [16], (*b*) they apply to two data sources instead of one single relation, and (*c*) they are used in data sources that may depend on ontologies and, if so, can be logically interpreted.

Thus, the formal context encoding the key extraction problem [6] has to be extended to deal with non functional properties in the appropriate way. Dependencies between classes through properties may entail dependencies between link keys, as checking the equality of two relations will rely on other link keys to be extracted. Such dependencies between keys must be accounted for in the definition of the formal context and will induce constraints on the globally extracted solution. Moreover, if dependencies involve cycles, extraction techniques able to handle such cycles are needed. For this reason, we resort to techniques developed for Relational Concept Analysis (RCA [40]) where relations between objects can be explicitly handled. However, RCA aims at identifying and characterising concepts within data, though our goal is to identify link keys across two datasets. Hence, the framework has to be adapted to this end with specific scaling operators.

In this paper, we first show how our link key candidate extraction algorithm [5] can be redefined as a formal concept analysis problem. We then show how this formulation can be extended to dependent link key candidate extraction and how relational concept analysis can be used to deal with circular dependencies.

These are useful results for developers of link key extraction systems: they provide a way to extract coherent families of link key candidates directly from datasets. No such algorithm was available before. Moreover, this is expressed through the principled extension of the well-studied formal concept analysis framework, and not through ad hoc algorithms.

Additionally, for FCA researchers, this work illustrates the use of the principles of relational concept analysis as a general-purpose mechanism, beyond the extraction of concept descriptions.

The outline of the paper is as follows. First, we discuss related work on the topics of data interlinking, key extraction and especially those developed with FCA (§2). Then, we introduce the problem and notations used in the paper (§3) as well as the basics of formal and relational concept analysis (§4). The encoding of the simple link key candidate extraction for RDF considered in [5] within the FCA framework is extended to non functional properties (§5). We show that the extracted formal concepts correspond to the expected link key candidates. This is extended to encompass dependent link keys and the notion of a coherent family of link key candidates is introduced to express the constraints spanning across link key candidates (§6). Then, we provide the RCA encoding of the problem of cyclic dependent link key candidate extraction through the definition of a relational context family and specific scaling operators (§7). Finally, we report on a proof-of-concept implementation of the approach that has been used throughout the paper to automatically provide the results of examples (§8).

2. Related work

Data interlinking refers to the process of finding pairs of IRIs used in different RDF datasets representing the same entity [21, 14, 34]. The result of this process is a set of links, which may be added to both datasets

by relating the corresponding IRIs with the owl:sameAs property. The task can be defined as: given two sets of individual identifiers I_D and $I_{D'}$ from two datasets D and D', find the set L of pairs of identifiers $\langle o, o' \rangle \in I_D \times I_{D'}$ such that $o = o'$.

Data interlinking is usually performed by using a framework, such as SILK [49] and LIMES [35], for processing *link specifications* that produce links. Link specifications indicate what are the conditions for two IRIs to be linked. They may be directly defined by users or automatically extracted.

This paper is concerned with the problem of automatically extracting a specific type of link specification from data. There are different types of link specifications. We distinguish between numerical and logical specifications.

Most methods roughly compute a numerical specification $\langle \sigma, \theta \rangle$ made of a similarity measure σ between the entities to be linked and a threshold θ. They assume that if two entities are very similar, they are likely the same. Hence, such specifications may generate links through (adapted from [44]):

$$L^{D,D'}_{\sigma,\theta} = \{\langle o, o' \rangle \in I_D \times I_{D'}; \sigma(o, o') \geq \theta\}$$

These numerical specifications are well adapted when approximate matches may refer to the same entities. Various methods have been designed for extracting numerical specifications based on spatial techniques [43], probabilities [46], or genetic programming and active learning [36]. Such numerical specifications may be extracted by using machine learning [27, 44]. A larger selection of methods is available in [34].

Logical link specifications are logical axioms from which the links are consequences. Logical specifications in general, and link keys in particular, are well-adapted when datasets offer enough ground for object identifying features, i.e. the descriptions of the same entity are likely an exact match on such features. Because of their logical interpretation, they are prone to be processed by logical reasoners or rule interpreters. There are various ways to approach the definition of such specifications [42, 3, 2, 25]. This paper deals with the extraction of link keys, a specific type of logical link specification. *Link keys* may be thought of as the generalisation of database keys to the case of two different datasets and to the specifics of RDF [19, 5].

Databases generated work on record linkage —or data deduplication— and keys [18]. They may be seen as analogous to the numerical and logical approaches to specify identity within the same dataset. Recently, data matching has been introduced as the problem of matching entities from two databases [14], but the use of keys was not considered. A key for a relation is a set of attributes which uniquely identifies one entity. Hence, two tuples with the same values for these attributes represent the same entity, and they are usually forbidden. Techniques for extracting keys, and more generally functional dependencies, from databases have been designed [45, 26]. Using lattices is common place for extracting functional dependencies [30, 17, 31]. The problem was considered in Formal Concept Analysis (FCA) in a functional setting [23] and further refined in pattern structures, the extension of FCA to complex data [9, 15].

There are two important differences, relevant to the definition of link specifications, between the data model used in databases, especially relational databases, and RDF:

- Multiple property values: In the relational model, attributes are functional, i.e. they bear a single value. In RDF, this is not the case, unless specified otherwise: an object may have several values for the same property. Hence, property values may be compared in different ways: either by considering that objects share at least one value for a property (IN-condition), or that they share all of them (EQ-conditions) [4].
- Object references: RDF is made of relations between entities, hence the values of a property may be objects. In the relational model, properties reach a value. In consequence, the database keys can establish the identity of property values through data equality, though RDF keys may have to compare objects, which requires, in turn, a key to be identified. This makes keys eventually dependent on each others.

Extensions of database keys addressing these issues have been provided for description logic languages [13, 32]. Several key extraction algorithms have been designed [7, 38, 47, 48] which extract key candidates from RDF datasets and select the most accurate key candidate according to key quality measures. Key extraction algorithms discover either IN-keys (only made of IN-conditions) [47, 1, 20] or EQ-keys (only made of EQ-conditions) [7], however though they can identify entities from the same dataset, they cannot do it across datasets unless these are using a common ontology or there exists an alignment between their ontologies.

The approaches proposed in [1, 20] aim at using a key extraction algorithm [47] to extract pairs of keys that can be used as link specifications. They extract IN-keys that hold in both source and target datasets. It is assumed that both datasets are described using the same ontology or, more precisely, the system only looks for keys based on the vocabulary common to the two datasets. In this case, discovered IN-keys, although not equal, mostly correspond to strong IN-link keys (link keys made up of keys) [6], and not to weak IN-link keys (more general), which are the kind of link keys extracted in [5].

There is no necessary correspondence between keys and link keys: there may exist keys which are part of no link key and link keys which do not rely on keys [19, Example 5.38, p. 116] and [6]. Hence, looking for keys to be eventually turned into link keys may fail to solve the problem.

Moreover, the types of keys considered so far can only be used as link specification as long as the datasets use the same classes and properties, at least for these keys. It is possible to deal with this problem by using an alignment between the ontologies of the two datasets, but the problem to solve is different.

In [5], we have proposed to directly discover link keys between two classes from two datasets. The proposed algorithm does not require an initial alignment between properties of both datasets and avoids the generation of keys that are specific to only one dataset. It first extracts link key candidates from the data and then uses measures of the quality of these candidates in order to select the one to apply. To select the best link key candidates, we have proposed two pairs of quality measures. The first ones, precision and recall, are supervised, i.e. they require both positive and negative examples of links. The second pair of measures, discriminability and coverage, are unsupervised, i.e. they do not require any link as input.

Direct link key candidate extraction with FCA has been described for relational databases [6]. However, it is defined for functional properties and must be adapted to the cases of multivalued and relational attributes. Moreover, it works on pairs of tables and does not extract dependent link key candidates, e.g. link keys of the class Book, which features the creator relation, will depend on link keys of the class Person.

There is no intrinsic superiority of one type of link specification, numerical or logical. They rather have to be used in a complementary way since the effectiveness of different methods may depend on the particular datasets. In general, key-based specifications are more likely to achieve higher precision, but lower recall than similarity-based specification. Additionally, the approach described here will deal with dependent link keys and does not need an alignment, though the approach of KeyRanker [20] requires an alignment and can treat dependencies through dealing with property paths.

In the following, we define how formal concept analysis can be exploited to extract dependent link key candidates from heterogeneous datasets, even in the presence of circular dependencies.

3. RDF datasets and link keys

In this section, we introduce preliminaries on RDF datasets and link keys used throughout the paper.

3.1. RDF datasets

Link keys are used to interlink datasets. In this paper, we focus on RDF datasets[2]. In RDF, resources are identified by Internationalized Resources Identifiers (IRIs) [16]. An RDF statement is a triple $\langle s, p, o\rangle$ where s, p and o are called the *subject*, *predicate* (or *property*) and *object* of the statement. The subject and predicate are resources; the object may be a resource or a literal, i.e. a value depending on a datatype. For instance, in Figure 1, the triple $\langle$`o1:z1`,`o1:firstname`,`"Thomas"`$\rangle$ has a literal as object and, $\langle$`o1:z1`,`rdf:type`,`o1:Person`$\rangle$ has a resource as object. In addition, RDF allows to declare anonymous resources using blank nodes, which act as existential variables, e.g. $\langle$`o1:z1`,`o1:hasAge`,`?x`$\rangle$. An RDF dataset is a set of RDF triples that can be viewed as a directed labelled multigraph. The subject and the object of each triple are labels of two nodes connected by an edge directed from the subject to the object, the edge being labelled with the predicate of the triple. Below we provide the definition of an RDF dataset.

Definition 1 (RDF dataset)**.** *Let U be a set of IRIs, B a set of blank nodes and L a set of literals. An* RDF dataset *is a set of triples from $(U \cup B) \times U \times (U \cup B \cup L)$.*

Given an RDF dataset D, we denote by U_D, B_D and L_D, respectively, the sets of IRIs, blank nodes and literals present in D.

A special property, very often used in RDF datasets, is `rdf:type`, declaring that an individual belongs to a particular class, e.g. $\langle$`o1:z1`,`rdf:type`,`o1:Person`$\rangle$. The following definition makes explicit the distinction between individuals, properties and classes of an RDF dataset.

Definition 2 (Identifiers of an RDF dataset)**.** *Let D be an RDF dataset, the sets I_D of* individual identifiers*, P_D of* datatype property identifiers*, R_D of* object property identifiers *and C_D of* class identifiers *in D, are defined as follows:*

– *$o \in I_D$ if and only if $o \in U_D$ and there exist p and u such that $\langle o, p, u\rangle \in D$ or $\langle u, p, o\rangle \in D$.*

[2] We use here the term RDF dataset instead of the standard term RDF graph [16] as these RDF graphs are not precisely graphs.

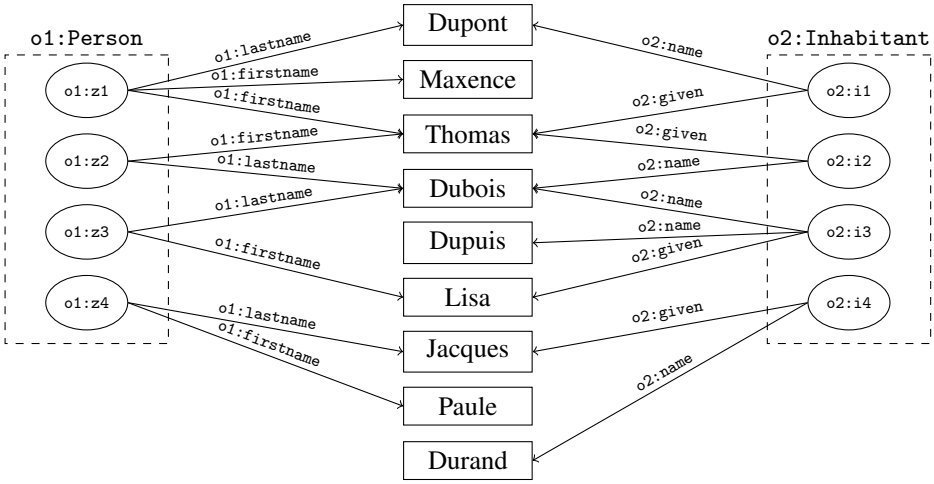

Figure 1: Example of two datasets representing respectively instances of classes `Person` and `Inhabitant`.

- $p \in P_D$ *if and only if* $p \in U_D$ *and there exist* o *and* u *such that* $u \in L_D$ *and* $\langle o, p, u\rangle \in D$.
- $r \in R_D$ *if and only if* $r \in U_D$ *and there exist* o *and* u *such that* $u \in U_D \cup B_D$ *and* $\langle o, r, u\rangle \in D$.
- $c \in C_D$ *if and only if* $c \in U_D$ *and there exists* o *such that* $\langle o, \texttt{rdf:type}, c\rangle \in D$.

The vocabulary of "class", "datatype property", "object property" and "individual" is used according to their meaning in RDFS [12] and OWL [33], but this does not need to be further specified for the sake of this paper. These sets may be assumed to be disjoint without loss of generality by separating different manifestations of the same symbol (as class, as datatype property, as object property, as individual). From these we define the signature of a dataset.

Definition 3 (Signature of an RDF dataset)**.** *The* signature *of an RDF dataset* D *is the tuple* $\langle R_D, P_D, C_D\rangle$.

We denote by $c^D = \{t \in I_D | \langle t, \texttt{rdf:type}, c\rangle \in D\}$ the set of instances of $c \in C_D$ in the dataset D. In RDF, an individual may have several different values for the same property. For a datatype property $p \in P_D$, we denote by $p^D(o) = \{v \in L_D | \langle o, p, v\rangle \in D\}$ the set of values of property p for object o in the dataset D. Similarly, for the object property $r \in R_D$, we have $r^D(o) = \{u \in I_D | \langle o, r, u\rangle \in D\}$.

Example 1 illustrates the notion of a dataset.

Example 1 (RDF dataset)**.** *Figure 1 shows an example of two simple datasets (`o1` and `o2`) to be interlinked. These datasets respectively represent instances of classes `Person` and `Inhabitant`. The first dataset describes persons using properties `lastname` and `firstname`. The second dataset makes use of properties `name` and `given` to describe inhabitants. All these properties are datatype properties. Hence, the signature of `o1` is* $\langle\{\}$ $\{$`o1:lastname`, `o1:firstname`$\}$ $\{$`o1:Person`$\}\rangle$ *and that of `o2` is* $\langle\{\}$ $\{$`o2:name`, `o2:given`$\}$ $\{$`o2:Inhabitant`$\}\rangle$.

Each dataset is populated with four instances. In the first dataset, there are instances `z1`, `z2`, `z3`, and `z4`. In the second dataset, instances are `i1`, `i2`, `i3`, and `i4`.

For example, the first dataset states that `z1` is an instance of `Person` who has `lastname` "Dupont" and `firstname` "Thomas" and "Maxence". This example shows that properties can be multivalued: `z1` has two firstnames and `i3` two names. In this example, we assume that the following links have to be found: `z1` = `i1`, `z2` = `i2`, and `z3` = `i3`. Instances `z4` and `i4` are obviously different.

3.2. Link keys

Link keys specify the pairs of properties to compare for deciding whether individuals of two classes of two different datasets have to be linked. We first give the definition of a link key expression.

Definition 4 (Link key expression)**.** *A* link key expression *over two signatures* $\langle R, P, C\rangle$ *and* $\langle R', P', C'\rangle$ *is an element of* $2^{(P\times P')\cup(R\times R')} \times 2^{(P\times P')\cup(R\times R')} \times (C \times C')$*, i.e.*

$$\langle \{\langle p_i, p'_i\rangle\}_{i\in EQ}, \{\langle q_j, q'_j\rangle\}_{j\in IN}, \langle c, c'\rangle\rangle$$

such that EQ and IN are (possibly empty) finite sets of indices.

In this section, we do not take into account the object property part of signatures which will be considered in Section 6.

In order to make the notation more legible, sometimes we will write:

$$\{\langle p_i, p_i'\rangle\}_{i\in EQ}\ \{\langle q_j, q_j'\rangle\}_{j\in IN}\ \text{linkkey}\ \langle c, c'\rangle$$

The two sets of conditions are respectively called $\forall$-conditions and $\exists$-conditions to distinguish them from the former EQ- and IN-conditions [4]: IN-conditions are the same as $\exists$-conditions, but EQ-conditions correspond to both $\forall$- and $\exists$-conditions.

Link key expressions may be compared and combined through subsumption, meet or join.

Definition 5 (Subsumption, meet and join of link key expressions)**.** *Let $K = \langle E, I, \langle c, c'\rangle\rangle$ and $H = \langle F, J, \langle c, c'\rangle\rangle$ be two link key expressions over the same pair of signatures $\langle R, P, C\rangle$ and $\langle R', P', C'\rangle$. We say that K* is subsumed by *H, written $K \trianglelefteq H$, if $E \subseteq F$ and $I \subseteq J$. Additionally, the* meet *and* join *of K and H, denoted by $K \triangle H$ and $K \triangledown H$, respectively, are defined as follows:*

$$K \triangle H = \langle E \cap F, I \cap J, \langle c, c'\rangle\rangle$$
$$K \triangledown H = \langle E \cup F, I \cup J, \langle c, c'\rangle\rangle$$

For any finite set of link key expressions over the same pair of signatures, their meet and join are well-defined and each element of the set subsumes the meet and is subsumed by the join ($K \triangle H \trianglelefteq K \trianglelefteq K \triangledown H$).

Notation: we write $K \triangleleft H$ when $K \trianglelefteq H$ and not $H \trianglelefteq K$ ($H \not\trianglelefteq K$).

We only consider subsumption, meet and join of link key expressions over the same pair of classes and the same pair of signatures. When two link key expressions are defined over different pairs of signatures, the union of the signatures can be used to provide a common pair of signatures.

So far, link key expressions have been defined over signatures independently of actual datasets: they are only syntactic expressions. Intuitively, a link key expression $\langle\{\langle p_i, p_i'\rangle\}_{i\in EQ}, \{\langle q_j, q_j'\rangle\}_{j\in IN}, \langle c, c'\rangle\rangle$ generates a link, denoted by $\langle o, o'\rangle$, between two individuals o and o' of c and c', respectively, if o has the same values for p_i as o' has for p_i' (for each $i \in EQ$) and o and o' share at least one value for q_j and q_j' ($j \in IN$). More formally:

Definition 6 (Link set generated by a link key expression)**.** *Let D and D' be two datasets of signatures $\langle R, P, C\rangle$ and $\langle R', P', C'\rangle$, respectively. Let $K = \langle\{\langle p_i, p_i'\rangle\}_{i\in EQ}, \{\langle q_j, q_j'\rangle\}_{j\in IN}, \langle c, c'\rangle\rangle$ be a link key expression over their signature. The* link set generated by *K for D and D' is the subset $L_K^{D,D'} \subseteq c^D \times c'^{D'}$ defined as:*

$$\langle o, o'\rangle \in L_K^{D,D'} \text{ iff } \begin{cases} p_i^D(o) = p_i'^{D'}(o') & \text{for all } i \in EQ, \text{ and} \\ q_j^D(o) \cap q_j'^{D'}(o') \neq \emptyset & \text{for all } j \in IN \end{cases}$$

If no confusion arises, we may write L_K instead of $L_K^{D,D'}$.

Example 2 illustrates the use of all the definitions introduced so far.

Example 2 (Link key expressions and link sets)**.** *Figure 1 shows an example of two simple datasets (`o1` and `o2`) to be interlinked. From their signature, it is possible to define the following link key expressions:*

$$K_1 = \langle\{\},\ \{\langle \texttt{o1:firstname}, \texttt{o2:given}\rangle\},\ \langle \texttt{o1:Person}, \texttt{o2:Inhabitant}\rangle\rangle$$
$$K_2 = \langle\{\},\ \{\langle \texttt{o1:lastname}, \texttt{o2:name}\rangle\},\ \langle \texttt{o1:Person}, \texttt{o2:Inhabitant}\rangle\rangle$$
$$K_3 = K_1 \triangledown K_2 = \langle\{\},\ \{\langle \texttt{o1:firstname}, \texttt{o2:given}\rangle, \langle \texttt{o1:lastname}, \texttt{o2:name}\rangle\},\ \langle \texttt{o1:Person}, \texttt{o2:Inhabitant}\rangle\rangle$$
$$K_4 = \langle\{\langle \texttt{o1:firstname}, \texttt{o2:given}\rangle, \langle \texttt{o1:lastname}, \texttt{o2:name}\rangle\},\ \{\},\ \langle \texttt{o1:Person}, \texttt{o2:Inhabitant}\rangle\rangle$$
$$K_5 = \langle\{\langle \texttt{o1:lastname}, \texttt{o2:given}\rangle\},\ \{\},\ \langle \texttt{o1:Person}, \texttt{o2:Inhabitant}\rangle\rangle$$

Such link key expressions would generate the following link sets:

$$L_{K_1}^{o1,o2} = \{\langle \texttt{o1:z1}, \texttt{o2:i1}\rangle, \langle \texttt{o1:z2}, \texttt{o2:i2}\rangle, \langle \texttt{o1:z3}, \texttt{o2:i3}\rangle, \langle \texttt{o1:z1}, \texttt{o2:i2}\rangle, \langle \texttt{o1:z2}, \texttt{o2:i1}\rangle\}$$
$$L_{K_2}^{o1,o2} = \{\langle \texttt{o1:z1}, \texttt{o2:i1}\rangle, \langle \texttt{o1:z2}, \texttt{o2:i2}\rangle, \langle \texttt{o1:z3}, \texttt{o2:i3}\rangle, \langle \texttt{o1:z3}, \texttt{o2:i2}\rangle, \langle \texttt{o1:z2}, \texttt{o2:i3}\rangle\}$$
$$L_{K_3}^{o1,o2} = L_{K_1}^{o1,o2} \cap L_{K_2}^{o1,o2} = \{\langle \texttt{o1:z1}, \texttt{o2:i1}\rangle, \langle \texttt{o1:z2}, \texttt{o2:i2}\rangle, \langle \texttt{o1:z3}, \texttt{o2:i3}\rangle\}$$
$$L_{K_4}^{o1,o2} = \{\langle \texttt{o1:z2}, \texttt{o2:i2}\rangle\}$$
$$L_{K_5}^{o1,o2} = \{\langle \texttt{o1:z4}, \texttt{o2:i4}\rangle\}$$

The difference between ∃-conditions and ∀-conditions is visible in the set of links generated by K_3 and K_4. In the latter case, all property values should be in both instances for being linked; in the former, one common value is sufficient. The link key expression that would generate the expected links for Example 1 is K_3.

With respect to link generation, subsumption, meet and join behave like set inclusion, intersection and union as the ∀- and ∃-conditions are independent. The following lemma makes explicit the relations between these operations and the generated link sets.

Lemma 1. *Let $K, H, K_1, \ldots, K_n$ be link key expressions for two datasets D and D' over the same pair of classes. The following holds (omitting the mention of the two datasets):*

$$\text{If } K \trianglelefteq H \text{ then } L_H \subseteq L_K \tag{1}$$

$$L_{\triangle_{k=1}^{n} K_k} \supseteq \bigcup_{k=1}^{n} L_{K_k} \tag{2}$$

$$L_{\triangledown_{k=1}^{n} K_k} = \bigcap_{k=1}^{n} L_{K_k} \tag{3}$$

Proof. In order to prove this lemma, we introduce the notion of the satisfaction of a link key condition. We say that a link $\langle o, o' \rangle$ satisfies the ∀-conditions $\{\langle p_i, p'_i \rangle\}_{i \in EQ}$ in datasets D and D' if $\forall i \in EQ, p_i^D(o) = p_i'^{D'}(o')$ and that it satisfies the ∃-conditions $\{\langle q_j, q'_j \rangle\}_{j \in IN}$ in datasets D and D' if $\forall j \in IN, q_j^D(o) \cap q_j'^{D'}(o') \neq \varnothing$. This is noted $\langle o, o' \rangle \propto \{\langle p_i, p'_i \rangle\}_{i \in EQ}$ and $\langle o, o' \rangle \propto \{\langle q_j, q'_j \rangle\}_{j \in IN}$ (omitting the mention of the two datasets).
(1) If $K \trianglelefteq H$, with $K = \langle E, I, \langle c, c' \rangle \rangle$, it can be assumed that $H = \langle E \cup F, I \cup J, \langle c, c' \rangle \rangle$. Hence, $\forall l \in L_H$, $l \propto E \cup F$ and $l \propto I \cup J$, thus $l \propto E$ and $l \propto I$ which means that $l \in L_K$. Therefore, $L_H \subseteq L_K$.
(2) $l \in \bigcup_{k=1}^{n} L_{K_k} \Leftrightarrow \exists k \in 1..n; l \in L_{K_k} \Leftrightarrow \exists k \in 1..n; l \propto E_k \wedge l \propto I_k \Rightarrow l \propto \bigcap_{k=1}^{n} E_k \wedge l \propto \bigcap_{k=1}^{n} I_k \Leftrightarrow l \in L_{\triangle_{k=1}^{n} K_k}$. Hence, $\bigcup_{k=1}^{n} L_{K_k} \subseteq L_{\triangle_{k=1}^{n} K_k}$.
(3) $l \in \bigcap_{k=1}^{n} L_{K_k} \Leftrightarrow \forall k \in 1..n, l \in L_{K_k} \Leftrightarrow \forall k \in 1..n, l \propto E_k \wedge l \propto I_k \Leftrightarrow l \propto \bigcup_{k=1}^{n} E_k \wedge l \propto \bigcup_{k=1}^{n} I_k \Leftrightarrow l \in L_{\triangledown_{k=1}^{n} K_k}$. Hence, $\bigcap_{k=1}^{n} L_{K_k} = L_{\triangledown_{k=1}^{n} K_k}$. □

Property 1 shows that if several link keys generate the same links, there is a link key that subsumes them and generates the same set of links.

Property 1. *The join of a set of link key expressions generating the same set of links, generates that same set of links.*

Proof. This is a direct consequence of Lemma 1(3). □

The capability of link key expressions to generate links across datasets, i.e. as link specifications, should now be clear. What is more difficult is to find the link key expression that generates the correct links. Of course, in principle, we do not know in advance which links are correct or not. We developed an algorithm which searches for link key *candidates* first before selecting the most promising candidate based on statistical evaluation [5]. Link key candidates are those link key expressions that, given a pair of datasets, would produce unique sets of links.

The definition of a link key candidate that we will use in this paper extends the one that we introduced in [5] by closing them by join. The link key candidates considered in [5] are now called "base link key candidates". Such a *base link key candidate* is a link key expression that generates at least one link $\langle o, o' \rangle$, and that is maximal among, i.e. subsumes, all the other link key expressions (over the same pair of classes) that also generate $\langle o, o' \rangle$. The maximal link key expression is the only normal form identifying a unique set of generated links. It may cover many non maximal link key expressions generating the same links, several of which may be minimal.

Definition 7 (Base link key candidate)**.** *Let D and D' be two datasets and K a link key expression over their signatures, K is a* base link key candidate *for D and D' if*

A1. there exists $l \in L_K^{D,D'}$, and

A2. if H is another link key expression over the same pair of classes as K such that $l \in L_H^{D,D'}$ and $K \trianglelefteq H$ then $K = H$.

A *link key candidate* is defined as a link key expression that generates at least one link and that is maximal for all the other link key expressions that generate the same link set. The reasons to extend our previous definition [5] are that (*a*) it more directly shows the link with formal concept analysis, and (*b*) candidate link keys obtained by join, which were not caught by the previous definition, are sometimes the best candidates (as illustrated in the forthcoming Example 4).

Definition 8 (Link key candidate)**.** *Let D and D' be two datasets and K a link key expression over their signatures, K is a* link key candidate *for D and D' if*

B1. $L_K^{D,D'} \neq \emptyset$*, and*

B2. $K = \bigtriangledown_{H \in [K]} H$ *such that* $[K] = \{H \mid L_K^{D,D'} = L_H^{D,D'}\}$

Intuitively, the set of link key expressions can be quotiented by the link sets they generate. This forms a partition of the link key expressions. Link key candidates are the maximal elements of the classes of this partition (Property 1).

Lemma 2. *Base link key candidates are link key candidates.*

Proof. Given a link key candidate K, since $l \in L_K$, then $L_K \neq \emptyset$. Moreover, if K is a base link key candidate, it is maximal for a particular link $l \in L_K$, i.e. there does not exist a link key expression H such that $K \lhd H$ (and thus not generating other links than those of L_K, because L is antimonotonic, Lemma 1(1)) that generates this link. This means that K is maximal for L_K. Hence, K is a link key candidate. □

Lemma 3 (The set of link key candidates is closed by meet)**.** *The meet of every finite set of link key candidates is a link key candidate.*

Proof. We need to prove that for any finite set $\{K_i\}_{i \in I}$ of link key candidates, $\bigtriangleup_{i \in I} K_i$ is a link key candidate. If for some $j \in I$; $K_j = \bigtriangleup_{i \in I} K_i$, then this is true. Considering that this is not the case, then necessarily, $\forall\, j \in I$, $L_{K_j} \neq L_{\bigtriangleup_{i \in I} K_i}$, otherwise, K_j would not be a link key candidate (not maximal). If $\bigtriangleup_{i \in I} K_i$ were not a link key candidate, this would mean that there exists H such that $L_{\bigtriangleup_{i \in I} K_i} = L_{(\bigtriangleup_{i \in I} K_i) \bigtriangledown H}$ and $H \not\trianglelefteq \bigtriangleup_{i \in I} K_i$. This entails that $\exists\, j \in I$, such that $l \in L_{K_j}$ and $l \notin L_{K_j \bigtriangledown H}$, hence $l \notin L_H$ (by Lemma 1(3)). Hence, $l \in L_{\bigtriangleup_{i \in I} K_i}$ (by Lemma 1(2)) and $l \notin L_{\bigtriangleup_{i \in I} K_i \bigtriangledown H}$ (by Lemma 1(3)), thus $L_{\bigtriangleup_{i \in I} K_i} \neq L_{(\bigtriangleup_{i \in I} K_i) \bigtriangledown H}$ which contradicts the hypothesis. □

The following proposition proves that being a link key candidate is equivalent to being a base link key candidate or the meet of base link key candidates.

Proposition 2. *The set of link key candidates is the smallest set containing all the base link key candidates that is closed by meet.*

Proof. (1) The set of link key candidates contains all base link key candidates (Lemma 2).

(2) The set of link key candidates is closed by meet (Lemma 3).

(3) The set of link key candidates is the smallest such set. If this were not the case, there would exist a link key candidate K not base and not obtained through the meet of link key candidates. There are two cases:

- either K is directly subsumed by the maximal link key candidate. Then, K generates at least one more link than the maximal link key candidate (otherwise it would not be a link key candidate because it would not be maximal for the link set it generates) and so K is a base link key candidate which contradicts the hypothesis.
- or there exists a set of link key candidates $\{K_i\}_{i \in I}$ subsuming K. This means that K is also subsumed by their meet ($\forall i \in I$, $K \lhd K_i$ entails $K \trianglelefteq \bigtriangleup_{i \in I} K_i$). Since, $K \neq \bigtriangleup_{i \in I} K_i$ by hypothesis, then $K \lhd \bigtriangleup_{i \in I} K_i$ and thus $L_{\bigtriangleup_{i \in I} K_i} \subset L_K$ (by Lemma 1(1) and because K would not be maximal if it would generate the same link set). Thus, there exists $l \in L_K \setminus L_{\bigtriangleup_{i \in I} K_i}$ and K is maximal for l (apparently no link key candidate that subsumes K generates l), hence K is a base link key candidate and this contradicts the hypothesis. □

Intuitively, this means that link key candidates are either base link key candidates or the meet of base link key candidates which is not maximal for any particular link (because for any link $l \in L_{\bigtriangleup_{i \in I} K_i}$, there exist i such as $l \in L_{K_i}$) but nonetheless generates a distinct set of links. Hence from the base link key candidates we can generate the set of link key candidates. However, we will show how to generate them directly with formal concept analysis.

4. A short introduction to FCA and RCA

We briefly introduce the principles of formal concept analysis and relational concept analysis which will be used to extract link key candidates.

4.1. Basics of formal concept analysis

Formal Concept Analysis (FCA) [23] starts with a binary context $\langle G, M, I \rangle$ where G denotes a set of objects, M a set of attributes, and $I \subseteq G \times M$ a binary relation between G and M, called the incidence relation. The statement gIm is interpreted as "object g has attribute m". Two operators $\cdot^{\uparrow}$ and $\cdot^{\downarrow}$ define a Galois connection between the powersets $\langle 2^G, \subseteq \rangle$ and $\langle 2^M, \subseteq \rangle$, with $A \subseteq G$ and $B \subseteq M$:

$$A^{\uparrow} = \{m \in M \mid gIm \text{ for all } g \in A\}$$
$$B^{\downarrow} = \{g \in G \mid gIm \text{ for all } m \in B\}$$

The operators $\cdot^{\uparrow}$ and $\cdot^{\downarrow}$ are decreasing, i.e. if $A_1 \subseteq A_2$ then $A_2^{\uparrow} \subseteq A_1^{\uparrow}$ and if $B_1 \subseteq B_2$ then $B_2^{\downarrow} \subseteq B_1^{\downarrow}$. Intuitively, the less objects there are, the more attributes they share, and dually, the less attributes there are, the more objects have these attributes. Moreover, it can be checked that $A \subseteq A^{\uparrow\downarrow}$ and that $B \subseteq B^{\downarrow\uparrow}$, that $A^{\uparrow} = A^{\uparrow\downarrow\uparrow}$ and that $B^{\downarrow} = B^{\downarrow\uparrow\downarrow}$.

For $A \subseteq G$, $B \subseteq M$, a pair $\langle A, B \rangle$, such that $A^{\uparrow} = B$ and $B^{\downarrow} = A$, is called a formal concept, where A is the extent and B the intent of $\langle A, B \rangle$. Moreover, for a formal concept $\langle A, B \rangle$, A and B are closed sets for the closure operators $\cdot^{\uparrow\downarrow}$ and $\cdot^{\downarrow\uparrow}$, respectively, i.e. $A^{\uparrow\downarrow} = A$ and $B^{\downarrow\uparrow} = B$.

Concepts are partially ordered by $\langle A_1, B_1 \rangle \leq \langle A_2, B_2 \rangle \Leftrightarrow A_1 \subseteq A_2$ or equivalently $B_2 \subseteq B_1$. With respect to this partial order, the set of all formal concepts forms a complete lattice called the concept lattice of $\langle G, M, I \rangle$.

Datasets are often complex with attribute values not ranging in Booleans, e.g. numbers, intervals, strings. Such data can be represented as a many-valued context $\langle G, M, W, I \rangle$, where G is a set of objects, M a set of attributes, W a set of values, and I a ternary relation defined on the Cartesian product $G \times M \times W$. The fact $\langle g, m, w \rangle \in I$ or simply $m(g) = w$ means that object g takes the value w for the attribute m. In addition, when $\langle g, m, w \rangle \in I$ and $\langle g, m, v \rangle \in I$ then $w = v$ [23]: in FCA, "many-valued" means that the range of an attribute may include more than two values, but for any object, the attribute can only have one of these values.

Conceptual scaling can be used for transforming a many-valued context into a one-valued context. For example, if $W_m = \{w_1, w_2, \ldots w_p\} \subseteq W$ denotes the range of the attribute m, then a scale of elements "$m = w_i, \forall w_i \in W_m$" is used for binarising the initial many-valued context. Intuitively, a scale splits the range W_m of a valued attribute m into a set of p binary attributes "$m = w_i, i = 1, \ldots, p$". There are many possible scalings and some of them are detailed in [23]. Another specific example of scaling is proposed hereafter for building relational attributes.

Conceptual scaling generally produces larger contexts than the original contexts —as the number of attributes increases. However, it is also possible to avoid scaling and to directly work on complex data, using the formalism of "pattern structures" [22, 28].

4.2. Relational concept analysis

Relational Concept Analysis (RCA) [40] extends FCA to the processing of relational datasets and allows inter-object relations to be materialised and incorporated into formal concept intents.

RCA is defined by a relational context family, composed of a set of formal contexts $\{\langle G_x, M_x, I_x \rangle\}_{x \in X}$ and a set of binary relations $\{r_y\}_{y \in Y}$. A relation $r_y \subseteq G_x \times G_z$ connects two object sets, a domain G_x ($dom(r_y) = G_x$, $x \in X$) and a range G_z ($ran(r_y) = G_z$, $z \in X$). The expression $r_y(g)$ denotes the set of objects in G_z related to g through r_y.

RCA is based on a *relational scaling* mechanism that transforms a relation r_y into a set of relational attributes that are added to the context describing the object set $dom(r_y)$, as made precise below. Various relational scaling operators exist in RCA such as existential, strict and wide universal, min and max cardinality, which all follow the classical role restriction semantics in description logics [8]. For example, strict universal scaling, consists of adding the attribute $\exists \forall r_y.C$ to M_x and to have $gI\exists \forall r_y.C$ whenever $r_y(g) \neq \varnothing$ and $r_y(g) \subseteq extent(C)$, i.e. whenever there exists an object related to g by r_y and all such objects belong to the extent of concept C.

Practically, in the case of strict universal scaling for example, let us consider two sets of objects, say G_x and G_z, their associated concept lattices L_x and L_z, and a relation $r_y \subseteq G_x \times G_z$. For each $C \in L_z$ for which there exists a pair of objects $g_x \in G_x$ and $g_z \in extent(C)$ such that $r_y(g_x, g_z)$ and for all $g_z \in r_y(g_x)$, $g_z \in extent(C)$, the process adds a new relational attribute $\exists \forall r_y.C$ to M_x.

The base RCA algorithm proceeds in the following way:

1. Initial formal contexts are for each $x \in X$, $\langle G_x, M_x^0, I_x^0 \rangle = \langle G_x, M_x, I_x \rangle$.
2. For each formal context $\langle G_x, M_x^t, I_x^t \rangle$ the corresponding concept lattice L_x^t is created using FCA.
3. Relational scaling is applied, for each relation r_y whose codomain lattice has new concepts, generating new contexts $\langle G_x, M_x^{t+1}, I_x^{t+1} \rangle$ including both plain and relational attributes in M_x^{t+1}.
4. If scaling has occurred ($\exists x \in X; M_x^{t+1} \neq M_x^t$), go to Step 2.
5. The results is a set of lattices $\{L_x^t\}_{x \in X}$.

The convergence of this process is detailed in [40] and formally proved in [41].

5. Formal contexts for independent link key candidates

In [6], the link key candidate extraction problem for databases was encoded in FCA. In this case, the set of database link key candidates exactly corresponds to the concepts of a one-valued context [23]. A concept $\langle L, K\rangle$ resulting from this encoding has as intent K a link key candidate and as extent L the link set it generates.

We generalise our encoding to the case of non functional datasets, such as RDF datasets, by adapting the link key conditions presented in Definition 4. These conditions deal with value multiplicity. The encoding of both conditions goes as follows, for each pair of properties $\langle p, p'\rangle \in P \times P'$, two attributes can be generated: $\exists\langle p, p'\rangle$ and $\forall\langle p, p'\rangle$ referred to, as before, as $\exists$-condition or $\forall$-condition.

Definition 9 (Formal context for independent link key candidates)**.** *Given two datasets D of signature $\langle R, P, C\rangle$ and D' of signature $\langle R', P', C'\rangle$, the* formal context for independent link key candidates *between a pair of classes $\langle c, c'\rangle$ of $C \times C'$ is $\langle c^D \times c'^{D'}, \{\exists, \forall\} \times P \times P', I\rangle$ such that:*

$$\langle o, o'\rangle \; I \; \forall\langle p, p'\rangle \;\; \textit{iff } p^D(o) = p'^{D'}(o') \tag{$\forall$}$$

$$\langle o, o'\rangle \; I \; \exists\langle p, p'\rangle \;\; \textit{iff } p^D(o) \cap p'^{D'}(o') \neq \varnothing \tag{$\exists$}$$

We introduce the $\forall\exists$ notation corresponding to:

$$\langle o, o'\rangle \; I \; \forall\exists\langle p, p'\rangle \;\; \text{iff } p^D(o) = p'^{D'}(o') \neq \varnothing \tag{$\forall\exists$}$$

which exactly corresponds to satisfying both $\forall$- and $\exists$-conditions, i.e. the values of the properties must be the same and there should be at least one such value. So they are only used as a shortcut when both ($\forall$) and ($\exists$) hold.

The Galois connection associated with such a formal context for link key candidates can be made explicit, as introduced in Section 4. Given two datasets D of signature $\langle R, P, C\rangle$ and D' of signature $\langle R', P', C'\rangle$ and a pair of classes $\langle c, c'\rangle \in C \times C'$, a Galois connection between the power sets $2^{(c^D \times c'^{D'})}$ and $2^{(\{\exists,\forall\}\times P\times P')}$ is defined as follows:

$$.^{\uparrow} : 2^{(c^D \times c'^{D'})} \longrightarrow 2^{(\{\exists,\forall\}\times P\times P')}$$

$$L^{\uparrow} = \{\exists\langle p, p'\rangle | \text{ for all } \langle o, o'\rangle \in L, \langle o, o'\rangle \; I \; \exists\langle p, p'\rangle\} \cup \{\forall\langle p, p'\rangle | \text{ for all } \langle o, o'\rangle \in L, \langle o, o'\rangle \; I \; \forall\langle p, p'\rangle\}$$

$$.^{\downarrow} : 2^{(\{\exists,\forall\}\times P\times P')} \longrightarrow 2^{(c^D \times c'^{D'})}$$

$$K^{\downarrow} = \{\langle o, o'\rangle | \text{ for all } \exists\langle p, p'\rangle \in K, \langle o, o'\rangle \; I \; \exists\langle p, p'\rangle \text{ and for all } \forall\langle p, p'\rangle \in K, \langle o, o'\rangle \; I \; \forall\langle p, p'\rangle\}$$

It should be clear from the definitions that for any pair of datasets D and D', $K^{\downarrow} = L_K^{D,D'}$.

Example 3 minimally illustrates the discovery of relevant link key candidates in FCA, but Example 4 is more elaborate.

Example 3 (Simple independent link key candidate extraction)**.** *Let us consider the two simple datasets given in Figure 2. We assume that classes `o1:Person` and `o2:Inhabitant` overlap, so we expect to find some link key candidates between these two classes. Following Definition 9, the formal context encoding the link key candidate extraction problem is built. This formal context is given in the table of Figure 3 (left), where instances `z2` and `i3` share at least one value, e.g. "Dubois", for the pair of properties `lastname` and `name`. In this simple example, since all properties are functional, $\forall$ and $\exists$ quantifications have identical columns. So, in the resulting lattice all the intent conditions are prefixed by $\forall\exists$. This leads to the concept lattice given in Figure 3 featuring five concepts. The concept $\langle\{\langle z_1, i_1\rangle, \langle z_2, i_2\rangle, \langle z_3, i_3\rangle\}, \{\forall\exists\,\langle$`lastname,name`$\rangle, \forall\exists\,\langle$`firstname,given`$\rangle\}\rangle$ whose intent represents the link key candidate $\{\forall\exists\,\langle$`lastname,name`$\rangle, \forall\exists\,\langle$`firstname,given`$\rangle\}$ would generate the links $\{\langle z_1, i_1\rangle, \langle z_2, i_2\rangle, \langle z_3, i_3\rangle\}$ (its extent) if used as a link key. This candidate is a perfect link key since it would generate all and only expected links. Other candidates that only use "lastname" or "firstname" would also generate all links but also wrong links since different persons may share either firstname or lastname.*

This encoding correctly conveys the notion of link key candidates as shown by the following proposition.

Proposition 3. *K is a link key candidate for datasets D and D' if and only if $\langle K^{\downarrow}, K\rangle$ is a formal concept of the formal context for link key candidates in D and D' and $K^{\downarrow} \neq \varnothing$.*

In other terms, this means that $\langle\{\langle p_i, p'_i\rangle\}_{i\in EQ}, \{\langle q_j, q'_j\rangle\}_{j\in IN}, \langle c, c'\rangle\rangle$ is a link key candidate for datasets D and D' if and only if $K = \{\forall\langle p_i, p'_i\rangle\}_{i\in EQ} \cup \{\exists\langle q_j, q'_j\rangle\}_{j\in IN}$ is the intent of a concept generated by the formal context for link key candidates between c and c' and $K^{\downarrow} \neq \varnothing$.

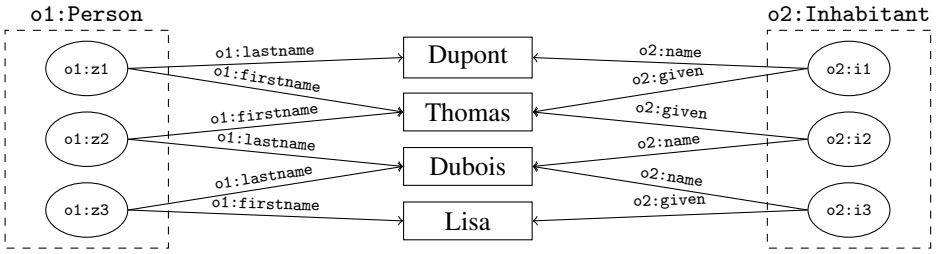

Figure 2: Example of two datasets representing respectively instances of classes `Person` and `Inhabitant`.

	$\forall\langle lastname, name\rangle$	$\forall\langle lastname, given\rangle$	$\forall\langle firstname, name\rangle$	$\forall\langle firstname, given\rangle$	$\exists\langle lastname, name\rangle$	$\exists\langle lastname, given\rangle$	$\exists\langle firstname, name\rangle$	$\exists\langle firstname, given\rangle$
$\langle z_1, i_1\rangle$	×			×	×			×
$\langle z_1, i_2\rangle$				×				×
$\langle z_1, i_3\rangle$								
$\langle z_2, i_1\rangle$				×				×
$\langle z_2, i_2\rangle$	×			×	×			×
$\langle z_2, i_3\rangle$	×				×			
$\langle z_3, i_1\rangle$								
$\langle z_3, i_2\rangle$	×				×			
$\langle z_3, i_3\rangle$	×			×	×			×

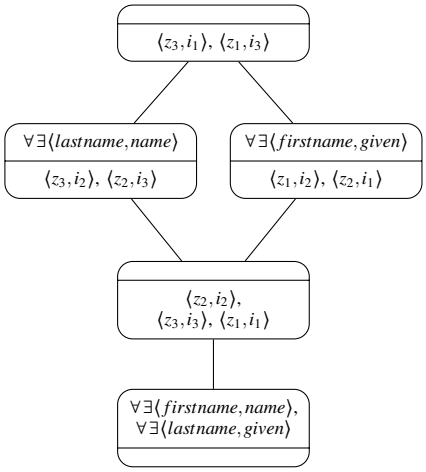

Figure 3: Formal context (left) and corresponding concept lattice (right) for the classes `o1:Person` and `o2:Inhabitant` of Figure 2.

Proof. The proof of the property is as follows. In both cases, $K^{\downarrow} = L_K \neq \varnothing$ (and we leave the D and D' implicit).

From Definition 8, K is a link key candidate if $L_K \neq \varnothing$ and $K = \nabla_{H \in [K]} H$ such that $[K] = \{H \mid L_K^{D,D'} = L_H^{D,D'}\}$.
($\Rightarrow$) Hence K is maximal for $L_K = K^{\downarrow}$ (Property 1), which means that $K^{\downarrow\uparrow} = K$, thus $\langle L_K, K\rangle$ is a concept.
($\Leftarrow$) Conversely, if $\langle K^{\downarrow}, K\rangle$ is a concept, $K^{\downarrow} = L_K$ and K is the maximal link key expression generating L_K, hence it is a link key candidate. □

Example 4 (Extraction of independent link key candidates over more complex data, as presented in Example 1)**.** *The datasets presented in Example 3 were very simple. Let us consider the dataset presented in Figure 1. The interlinking problem is now encoded in the context given in Table 1 and the resulting lattice is presented in Figure 4. The concepts corresponding to link key expressions of Example 2 are labelled by a +. Since some properties are multivalued, the lattice now contains some candidates with the same sets of properties but with different quantifications. For instance, there are two candidates whose intents are* $\{\exists\langle$ `firstname,given` $\rangle\}$ *and* $\{\forall\exists\langle$ `firstname,given` $\rangle\}$*. The former is less restrictive and then it would generate all the links generated by the latter plus the link* $\langle z_1, i_2\rangle$ *(because* p_1 *has the two firstnames "Maxence" and "Thomas" and* i_2 *has only "Thomas"). In this example, the perfect link key is labelled* $\star$ *whose intent is* $\{\exists\langle$ `firstname,given` $\rangle, \exists\langle$ `lastname,name` $\rangle\}$*. This is a link key candidate, but* not *a base link key candidate: it does not generate any link which is not generated by its subsumees.*

6. Hierarchically dependent link key candidates

So far, the proposed approach only dealt with datatype properties, i.e. properties whose value is not another object [16]. However, it may happen that the ranges of properties are themselves objects. These are called object properties or simply relations.

Determining if values of such properties across two datasets are equal or intersect requires to be able to identify them. This is exactly the role of link keys. Hence, link key candidates for pair of classes having object properties should rely on the link key candidates attached to the range of such properties.

	$\exists\langle firstname, name\rangle$	$\exists\langle firstname, given\rangle$	$\exists\langle lastname, name\rangle$	$\exists\langle lastname, given\rangle$	$\forall\langle firstname, name\rangle$	$\forall\langle firstname, given\rangle$	$\forall\langle lastname, name\rangle$	$\forall\langle lastname, given\rangle$
$\langle z_1, i_1\rangle$		×	×				×	
$\langle z_1, i_2\rangle$		×						
$\langle z_1, i_3\rangle$								
$\langle z_1, i_4\rangle$								
$\langle z_2, i_1\rangle$		×				×		
$\langle z_2, i_2\rangle$		×	×			×	×	
$\langle z_2, i_3\rangle$			×					
$\langle z_2, i_4\rangle$								
$\langle z_3, i_1\rangle$								
$\langle z_3, i_2\rangle$			×				×	
$\langle z_3, i_3\rangle$		×	×			×		
$\langle z_3, i_4\rangle$								
$\langle z_4, i_1\rangle$								
$\langle z_4, i_2\rangle$								
$\langle z_4, i_3\rangle$								
$\langle z_4, i_4\rangle$				×				×

Table 1: Formal context for the classes `o1:Person` and `o2:Inhabitant` of Figure 1.

Precisely, dealing with pairs of object properties $\langle r, r'\rangle$ whose range is the pair of classes $\langle d, d'\rangle$ requires to know how to compare the members of d and d' in order to decide if $r^D(o) = r'^{D'}(o')$ or if $r^D(o) \cap r'^{D'}(o') \neq \emptyset$. Comparison of sets of values in the definition of link sets generated by a link key candidate (Definition 6) and in the definition of the incidence relation (Definition 9) is achieved by using link key expressions on the range classes (Definition 10).

Definition 10 (Comparison of sets of instances)**.** *Given two sets S and S' of instances of classes d and d' and K a link key expression for the pair $\langle d, d'\rangle$, $=_K$ and $\cap_K$ are defined by:*

$$S =_K S' \quad \textit{iff} \quad \forall x \in S, \exists x' \in S'; \langle x, x'\rangle \in K^{\downarrow} \textit{ and } \forall x' \in S', \exists x \in S; \langle x, x'\rangle \in K^{\downarrow}$$
$$S \cap_K S' \neq \emptyset \quad \textit{iff} \quad K^{\downarrow} \cap (S \times S') \neq \emptyset$$

The use of these comparisons is only well-defined when there is no cyclic dependency across link key expressions.

This can be used in order to extend the formal contexts to dependent link key candidates:

Definition 11 (Formal context for dependent link key candidates)**.** *Given two datasets D of signature $\langle R, P, C\rangle$ and D' of signature $\langle R', P', C'\rangle$, for any pair of object properties $\langle r, r'\rangle \in R \times R'$, let $\mathcal{K}_{r,r'}$ the set of dependent link key candidates associated with the range of r and r'. The* formal context for dependent link key candidates *between a pair of classes $\langle c, c'\rangle$ of $C \times C'$ is $\langle c^D \times c'^{D'}, \{\forall, \exists\} \times ((P \times P') \cup \{\langle r, r', K\rangle; r \in R, r' \in R', K \in \mathcal{K}_{r,r'}\}), I\rangle$ such that:*

$$\langle o, o'\rangle \; I \; \forall\langle p, p'\rangle \;\; \textit{iff } p^D(o) = p'^{D'}(o')$$
$$\langle o, o'\rangle \; I \; \exists\langle p, p'\rangle \;\; \textit{iff } p^D(o) \cap p'^{D'}(o') \neq \emptyset$$
$$\langle o, o'\rangle \; I \; \forall\langle r, r'\rangle_K \;\; \textit{iff } r^D(o) =_K r'^{D'}(o')$$
$$\langle o, o'\rangle \; I \; \exists\langle r, r'\rangle_K \;\; \textit{iff } r^D(o) \cap_K r'^{D'}(o') \neq \emptyset$$

In principle, a link key is the way to identify the instances of a pair of classes. Hence, comparing the values of two properties should be done with the link key extracted for the classes of these values. For this reason, link key expressions (Definition 4) do not refer to the link keys to use as Definition 11. This means that the extracted

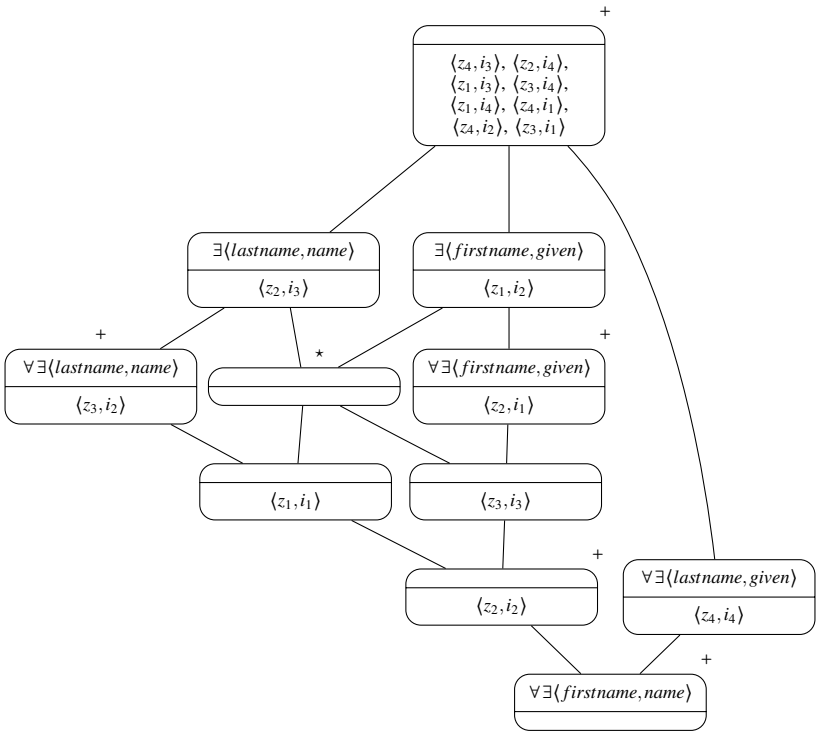

Figure 4: Concept lattice built from the context of Table 1 for the classes `o1:Person` and `o2:Inhabitant`.

link keys should only rely on each others. We aim at extracting such sets of link key candidates. We call them coherent families of link key candidates.

For simplifying the presentation we introduce the notion of alignment. An alignment in general is made of a set of relations between ontology entities [19]. For the purpose of this paper, it will simply be a set of pairs of classes considered as overlapping. A coherent set of link key candidates with respect to an alignment is called a coherent family of link key candidates.

Definition 12 (Coherent family of link key candidates)**.** *Given two datasets D of signature $\langle R,P,C\rangle$ and D' of signature $\langle R',P',C'\rangle$, and an alignment $A \subseteq C \times C'$ between their classes, a* coherent family of link key candidates *for D and D' related by A is a collection of dependent link key candidates:*

$$T = \{K_{\langle c,c'\rangle}\}_{\langle c,c'\rangle \in A}$$

such that each link key in T only depends on other link keys in T (compatibility).

Any collection of independent link key candidates with respect to an alignment A is a coherent family of link key candidates.

The key dependency graph of a coherent family of link key candidates is the directed graph whose vertices are the link key candidates and there is an edge from one link key candidate to another if the former depends on the latter. A coherent family of link key candidates is said to be "acyclic" if its key dependency graph contains no cycle, including unit cycle. Otherwise, it is called "cyclic".

When classes are hierarchically organised, coherent families of link key candidates are necessarily acyclic. They can be obtained inductively through the following algorithm:

1. Generate the (independent) link key candidates for pairs of classes having no object properties, using the formal contexts of Definition 9.
2. For each pair of classes that only depends on pairs of classes having been processed, build formal contexts according to Definition 11.
3. If there are still pairs of classes in A to process, go to Step 2.
4. Extract the coherent families by picking one link key candidate per pair of classes in A, as long as they are compatible.

Example 5 illustrates the extraction of such dependent link key candidates.

Example 5 (Acyclic link key candidate extraction)**.** *Figure 5, shows two datasets that both describe instances from classes* `House`*, resp.* `Place`*, that are in relation with instances from classes* `Person`*, resp.* `Inhabitant`*. First, the formal context between the two independent classes* `Person` *and* `Inhabitant` *is used to build the corresponding lattice, see Figure 6. Four link key candidates, named C0, C1, C2, C3, are extracted.*

From these candidates, it can be dealt with classes which only depend on the aligned pairs handled in the former steps, e.g. the pair of classes `Person` *and* `Inhabitant`*. The formal context for these two classes is given in Figure 7. Contrasting datatype properties, the pair* $\langle$`owner`, `ownedBy`$\rangle$ *(whose domains are classes* `House` *and* `Place`*) is taken into account according to the four link key candidates generated for classes* `Person` *and* `Inhabitant`*. The resulting lattice is given in Figure 7 using a reduced notation, i.e. when the intent of a concept contains several concepts computed at previous steps, only the most specific ones are kept. For example, concept D3 could also include* $\forall\exists\langle$`owner`, `ownedBy`$\rangle_{C0}$ *in its intent, but since C1 is more constraining than C0, then only* $\forall\exists\langle$`owner`, `ownedBy`$\rangle_{C1}$ *is retained.*

In the two lattices, compatible concepts are coloured: purple for C1 – D1 and C1 – D3, and green for C2 – D0 and C2 – D2. The green colour is reserved to the link key candidates that were expected; dotted pattern is used for candidates that are not part of any coherent family of link key candidates. The lattice shows that the intent of concept D0, i.e. $\{\forall\exists\langle$`owner`, `ownedBy`$\rangle_{C2}, \forall\exists\langle$`city`, `city`$\rangle\}$*, is the best link key candidate since it generates all and only expected links.*

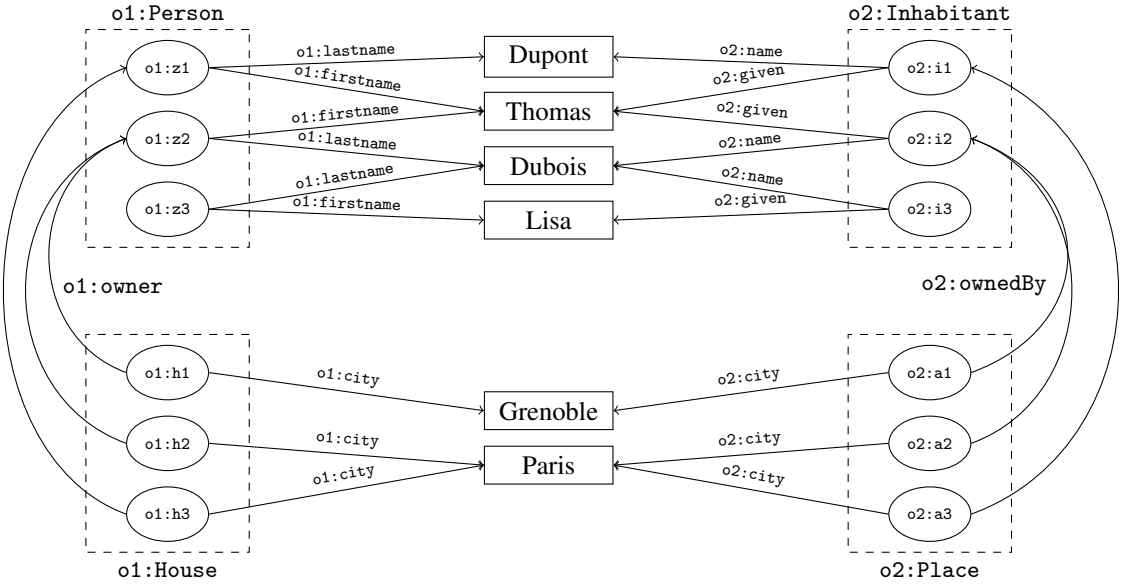

Figure 5: Two datasets representing instances of class `House` (resp. `Place`) that are in relation through the owner property, (resp. `ownedBy`) with instances of class `Person` (resp. `Inhabitants`).

.

A limitation of this method is that it cannot handle cases in which the data dependencies involve cycles. We now consider this case.

7. Dealing with cyclic dependencies through relational concept analysis

Classes are not always hierarchically organised. Object properties may loop through the set of classes. Even if this does not affect resulting link key candidates, this cannot be ruled out.

To solve this problem we took inspiration from RCA [40]. Accordingly, the problem is modelled, as in RCA, through the set of formal contexts corresponding to pairs of classes and a set of relational contexts corresponding to pairs of object properties. We thus diverge from the solutions provided in Section 6 by initially recording only datatype properties within the formal contexts (Definition 9) and using relational contexts for object properties.

The main difference with RCA is that RCA is designed for extracting conceptual descriptions of classes, though we aim at extracting link key candidates. The former describes classes, the latter describes how to identify instances. Another difference is that coherent families of link key candidates have no analogous notion in RCA as all class descriptions may coexist: no compatibility is requested. However, at the technical level of concept

	$\exists\langle firstname, given\rangle$	$\exists\langle firstname, name\rangle$	$\exists\langle lastname, given\rangle$	$\exists\langle lastname, name\rangle$	$\forall\langle firstname, given\rangle$	$\forall\langle firstname, name\rangle$	$\forall\langle lastname, given\rangle$	$\forall\langle lastname, name\rangle$
$\langle z_1, i_1\rangle$	×			×	×			×
$\langle z_1, i_2\rangle$	×				×			
$\langle z_1, i_3\rangle$								
$\langle z_2, i_1\rangle$	×				×			
$\langle z_2, i_2\rangle$	×			×	×			×
$\langle z_2, i_3\rangle$				×				×
$\langle z_3, i_1\rangle$								
$\langle z_3, i_2\rangle$				×				×
$\langle z_3, i_3\rangle$	×			×	×			×

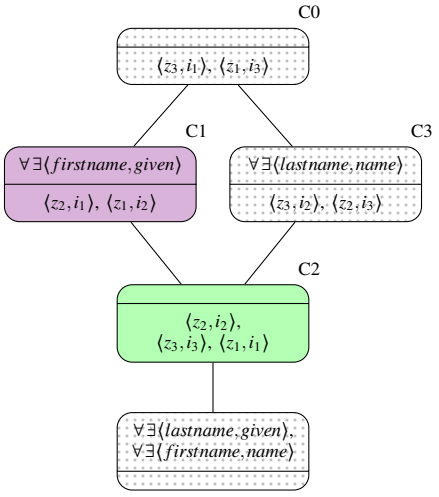

Figure 6: Formal context and concept lattice for `Person` and `Inhabitant` of Figure 5.

.

	$\exists\langle city, city\rangle$	$\forall\langle city, city\rangle$	$\exists\langle owner, ownedBy\rangle_{C0}$	$\forall\langle owner, ownedBy\rangle_{C0}$	$\exists\langle owner, ownedBy\rangle_{C1}$	$\forall\langle owner, ownedBy\rangle_{C1}$	$\exists\langle owner, ownedBy\rangle_{C2}$	$\forall\langle owner, ownedBy\rangle_{C2}$	$\exists\langle owner, ownedBy\rangle_{C3}$	$\forall\langle owner, ownedBy\rangle_{C3}$
$\langle h_1, a_1\rangle$	×	×	×	×	×	×	×	×	×	×
$\langle h_1, a_2\rangle$			×	×	×	×	×	×	×	×
$\langle h_1, a_3\rangle$			×	×	×	×				
$\langle h_2, a_1\rangle$			×	×	×	×	×	×	×	×
$\langle h_2, a_2\rangle$	×	×	×	×	×	×	×	×	×	×
$\langle h_2, a_3\rangle$	×	×	×	×	×	×				
$\langle h_3, a_1\rangle$			×	×	×	×				
$\langle h_3, a_2\rangle$	×	×	×	×	×	×				
$\langle h_3, a_3\rangle$	×	×	×	×	×	×	×	×	×	×

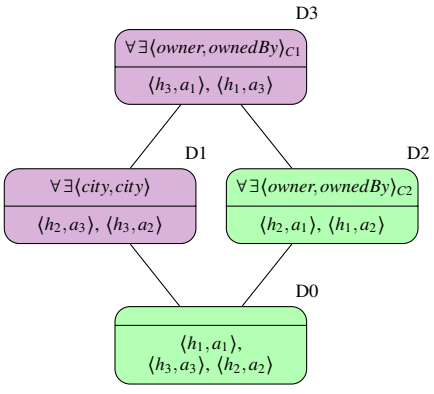

Figure 7: Dependent formal context and concept lattice for `House` and `Place` of Figure 5.

.

discovery, the process is the same. In particular, the relational contexts are exactly the same. They are simply extended to pairs of properties and thus pairs of instances.

Definition 13 (Relational contexts for dependent link key candidates)**.** *Given two datasets D of signature $\langle R,P,C\rangle$ and D' of signature $\langle R',P',C'\rangle$ and an alignment $A \subseteq C \times C'$, given two pairs of classes $\langle c,c'\rangle$ and $\langle d,d'\rangle$ of A and a pair of object properties $\langle r,r'\rangle$ of $R \times R'$, the relational context associated with r and r' in D and D' is defined by $\langle c^D \times c'^{D'}, d^D \times d'^{D'}, I\rangle$ such that:*

$$\langle x,x'\rangle \; I \; \langle y,y'\rangle \;\; \textit{iff} \;\; y \in r^D(x) \;\; \textit{and} \;\; y' \in r'^{D'}(x')$$

Scaling operators are used for transferring the constraints from the relational contexts to the formal contexts.

Definition 14 (Link key condition scaling operators)**.** *Given two datasets D of signature $\langle R,P,C\rangle$ and D' of signature $\langle R',P',C'\rangle$, and a relational context $\langle c^D \times c'^{D'}, d^D \times d'^{D'}, I\rangle$ for a pair of object properties $\langle r,r'\rangle \in R \times R'$,*

- *the universal scaling operator $f^{\forall}_{r,r'} : (c^D \times c'^{D'}) \times \mathcal{K}_{r,r'} \to \mathbb{B}$ is defined by $f^{\forall}_{r,r'}(\langle o,o'\rangle, K)$ iff $r^D(o) =_K r'^{D'}(o')$,*
- *the existential scaling operator $f^{\exists}_{r,r'} : (c^D \times c'^{D'}) \times \mathcal{K}_{r,r'} \to \mathbb{B}$ is defined by $f^{\exists}_{r,r'}(\langle o,o'\rangle, K)$ iff $r^D(o) \cap_K r'^{D'}(o') \neq \varnothing$.*

The universal and existential scaling operators scale the formal contexts in Definition 9 upon the $\forall$-conditions and $\exists$-conditions on the object property pairs of the relational contexts in Definition 13.

Hence, a relational context family [40] can be associated to two datasets and an alignment:

Definition 15 (Relational context family for dependent link key candidates)**.** *Given two datasets D of signature $\langle R,P,C\rangle$ and D' of signature $\langle R',P',C'\rangle$, and an alignment $A \subseteq C \times C'$, the relational context family for dependent link key candidate extraction between D and D' through A is composed of:*

- *all the formal contexts corresponding to pairs of classes in A,*
- *all the relational contexts corresponding to object property pairs in $R \times R'$ relating pairs of classes from A.*

The solution to the problem is a collection of concept lattices, one per pair of aligned classes in A, from which the coherent families of link key candidates may be extracted exactly like in Section 6. The algorithm is that of relational concept analysis, using the link key condition scaling operators (Definition 14):

1. Start with the formal contexts corresponding to aligned classes.
2. Apply formal concept analysis to all contexts.
3. Apply the scaling operators on the relational contexts to add columns in the formal contexts depending on the already extracted keys.
4. If scaling has occurred, go to Step 2.
5. Extract the coherent families by picking one link key candidate per pair of classes in A, as long as they are compatible.

Example 6 illustrates the extraction of cyclic link key candidates.

Example 6 (Cyclic link key candidate extraction)**.** *Figure 8 provides an example of two datasets containing circular dependencies. In the first, left-hand side, dataset, classes* `Person` *and* `House` *mutually depend on each other through relations* `home` *and* `owner`*. Respectively, in the second, right-hand side, dataset, classes* `Inhabitant` *and* `Place` *also mutually depend on each other through relations* `address` *and* `ownedBy`*. The relational context family is made of formal contexts presented in Figure 9 and relational contexts given in Figure 10. It is iteratively scaled using the scaling operators of Definition 14. It converges to a fix-point after six iterations. The final lattices are given in Figure 11. We obtain two pairs of compatible concepts, namely $C1-C2$ and $C4-C6$. The coherent family of link key candidates $C4-C6$ generates all and only relevant links.*

8. Implementation and complexity considerations

One benefit of reformulating the link key candidate extraction problem as formal and relational concept analysis is to use directly standard algorithms for such problems. We implemented the proposed approach as a proof-of-concept in Python 3[3]. The implementation took inspiration from another formal concept analysis implementation [39]. It uses the Norris algorithm [37] for performing FCA extended to deal with pairs of objects in the extent and quantified and qualified pairs of properties in the intent. The RCA processing has been implemented by iteratively

[3]The implementation is available from https://moex.inria.fr/software/linkky/.

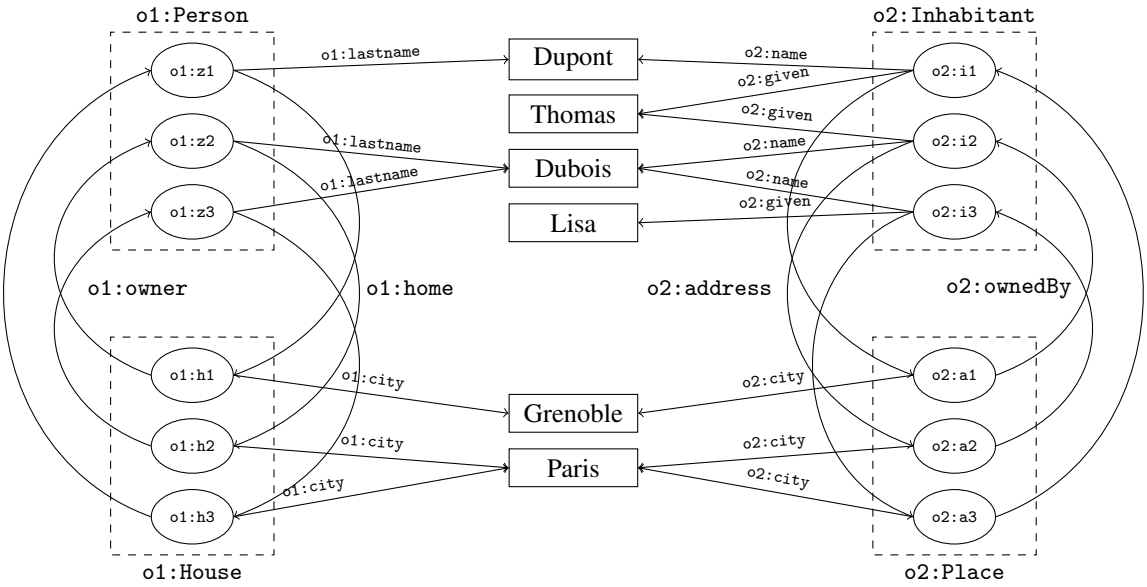

Figure 8: Datasets illustrating mutual dependencies between classes.

	$\exists\langle lastname, given\rangle$	$\exists\langle lastname, name\rangle$	$\forall\langle lastname, given\rangle$	$\forall\langle lastname, name\rangle$
$\langle z_1, i_1\rangle$		×		×
$\langle z_1, i_2\rangle$				
$\langle z_1, i_3\rangle$				
$\langle z_2, i_1\rangle$				
$\langle z_2, i_2\rangle$		×		×
$\langle z_2, i_3\rangle$		×		×
$\langle z_3, i_1\rangle$				
$\langle z_3, i_2\rangle$		×		×
$\langle z_3, i_3\rangle$		×		×

	$\exists\langle city, city\rangle$	$\forall\langle city, city\rangle$
$\langle h_1, a_1\rangle$	×	×
$\langle h_1, a_2\rangle$		
$\langle h_1, a_3\rangle$		
$\langle h_2, a_1\rangle$		
$\langle h_2, a_2\rangle$	×	×
$\langle h_2, a_3\rangle$	×	×
$\langle h_3, a_1\rangle$		
$\langle h_3, a_2\rangle$	×	×
$\langle h_3, a_3\rangle$	×	×

Figure 9: Formal contexts for the pairs of classes $\langle$`Person`, `Inhabitant`$\rangle$ and $\langle$`HousePlace`$\rangle$ in Figure 8.

$R_{owner,ownedBy}$	$\langle z_3, i_1\rangle$	$\langle z_3, i_2\rangle$	$\langle z_3, i_3\rangle$	$\langle z_2, i_1\rangle$	$\langle z_2, i_2\rangle$	$\langle z_2, i_3\rangle$	$\langle z_1, i_1\rangle$	$\langle z_1, i_2\rangle$	$\langle z_1, i_3\rangle$
$\langle h_1, a_1\rangle$					×				
$\langle h_1, a_2\rangle$						×			
$\langle h_1, a_3\rangle$				×					
$\langle h_2, a_1\rangle$		×							
$\langle h_2, a_2\rangle$			×						
$\langle h_2, a_3\rangle$	×								
$\langle h_3, a_1\rangle$								×	
$\langle h_3, a_2\rangle$									×
$\langle h_3, a_3\rangle$							×		

$R_{home,address}$	$\langle h_3, a_2\rangle$	$\langle h_3, a_3\rangle$	$\langle h_3, a_1\rangle$	$\langle h_2, a_2\rangle$	$\langle h_2, a_3\rangle$	$\langle h_2, a_1\rangle$	$\langle h_1, a_2\rangle$	$\langle h_1, a_3\rangle$	$\langle h_1, a_1\rangle$
$\langle z_1, i_1\rangle$									×
$\langle z_1, i_2\rangle$							×		
$\langle z_1, i_3\rangle$								×	
$\langle z_2, i_1\rangle$						×			
$\langle z_2, i_2\rangle$				×					
$\langle z_2, i_3\rangle$					×				
$\langle z_3, i_1\rangle$			×						
$\langle z_3, i_2\rangle$	×								
$\langle z_3, i_3\rangle$		×							

Figure 10: Relational contexts for pairs of relations $\langle$`owner`, `ownedBy`$\rangle$ and $\langle$`home`, `address`$\rangle$ in Figure 8.

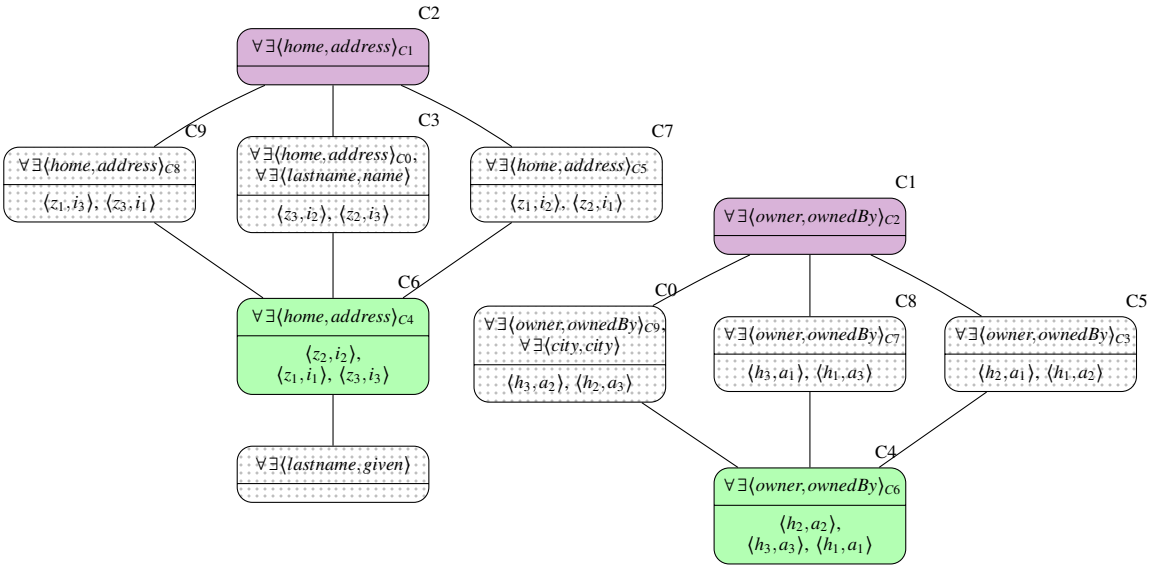

Figure 11: Concept lattices obtained from the relational context family of Figures 9 and 10 after six iterations.

applying the scaling operators to the formal contexts. RDF data can be loaded in the system through the RDF Library and lattices are plotted with Graphviz. The link key candidate extraction algorithm is more powerful than what is presented here as it does not rely at all on alignments. In addition, we implemented the unsupervised link key selection measures [5] and extended them to coherent families of link key candidates.

All examples presented above have been computed by the developed system. The formal and relational contexts and concept lattices have been directly generated by this implementation (we only changed node colours and patterns for legibility).

These examples only contain a few instances. We have tested the implemented software on the larger datasets that were used in [5] but run into scalability issues as no optimisation was implemented. However, we noticed that when considering samples of the datasets, the system was able to discover the correct link key candidates, and due to the nature of the unsupervised selection measure able to correctly identify the correct link key. More experiments must still be performed for supporting this claim with certainty.

The complexity of the proposed approach may be evaluated with respect to data or schema which are the two sides of our formal concepts. On the data side, we may count n_I as the number of individuals in one dataset. On the schema side, we may count n_C as the number of classes and n_P as the number of properties and relations. For the extraction of link keys for a pair of classes, the size of the input is, in the worst case, $|G| = n_I^2$ for the number of rows and $|M| = 2 \times n_P^2$ for the number of columns.

The upper bound to the number of concepts ($|L|$) is either the size of the power set of the set of all possible links or the number of link key expressions if smaller. The data upper bound of $2^{n_I^2}$ concepts can actually be achieved. The upper bound to the number of link key expressions in concepts is between $2^{n_P^2}$ and $4^{n_P^2}$ (not all link key expressions can generate a link but at least all IN-link key candidates can). We retain the highest figure. Thus, the size of the lattice $|L|$ is bounded by $min(2^{n_I^2}, 4^{n_P^2})$.

It is very likely that the worst-case complexity of our algorithms is that of FCA (or RCA). The complexity of formal concept extraction algorithms are given in [29] as between $O(|G|^2|M||L|)$ and $O(|G||M|^2|L|)$. Hence, if one considers that $n_P \ll n_I$, using the complexity of the Norris algorithm, we end up with $O(n_I^4 4^{n_P^2})$: an algorithm polynomial in the number of objects and exponential in the number of properties. This would not be practicable for an online algorithm, but for one-shot extraction it can be.

Using RCA, the number of relational attributes may be higher than $4^{n_P^2}$ since it depends on the number of concepts in the range lattices. It remains however bounded by $2^{n_I^2}$. Hence, the worst-case complexity should rather be exponential in $O(n_I^4 2^{n_I^2})$.

However, as very often, this worst-case complexity is not observed in practice. Previously, we observed that out of 1.9×10^{19} maximum link key candidates, we only had 17 link key candidates [5]. This was only for IN-link keys, actually for full link keys, out of 3.4×10^{38} link key expressions, we have 18 link key candidates. This

extraction process, with the non-FCA-based system, takes less than a minute on a 2.7GHz i7 laptop.

9. Conclusions

We have shown how FCA can be applied to the precise task of extracting links across RDF datasets and, more specifically, extracting link key candidates. For that purpose, specific contexts have been defined to express link key expressions as intent and adapted to pairs of datasets, dealing with pairs of classes, pairs of properties and pairs of instances. We showed how this formulation generalises our previous definitions [5]. This has been extended to simultaneously extract coherent families of dependent link key candidates. Finally, the techniques developed for RCA have been adapted to extract such families in the presence of circular dependencies. All results here apply as well to the extraction of (unary) key candidates and generalise straightforwardly to n-ary link key candidates.

So far, no algorithm had been provided for dealing with these problems but the first one. Defining them in the FCA and RCA framework yields a coherent and elegant formulation of all these problems that is directly processable. This is a useful model for developers of link key extraction systems. It is also an illustration of the general-purpose aspect of RCA techniques, usually achieved by introducing new scaling operators [11]. It is exploited here to extract link key candidates instead of concept descriptions.

These results are encouraging and a proof-of-concept system was developed. Although this system goes beyond what is presented here, it also suffers from scalability issues. We plan to improve on these issues through indexing and pruning techniques. We also plan to develop an approximate approach in which the presented process is performed several times on smaller samples of large datasets and then a consensus on the best link key candidates is established (using the measures of [5] on the full datasets).

Acknowledgements

This work has been partially supported by the ANR project Elker (ANR-17-CE23-0007-01, for the four first authors). Jérémy Vizzini has been partially supported by a grant from the PERSYVAL-Lab LabEx (ANR-11-LABX-0025-01) funded by the French "Investissement d'avenir" program.

References

[1] Manel Achichi, Mohamed Ben Ellefi, Danai Symeonidou, and Konstantin Todorov. Automatic key selection for data linking. In *Proc. 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW), Bologna (IT)*, volume 10024 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2016.

[2] Mustafa Al-Bakri, Manuel Atencia, Jérôme David, Steffen Lalande, and Marie-Christine Rousset. Uncertainty-sensitive reasoning for inferring sameAs facts in linked data. In *Proc. 22nd european conference on artificial intelligence (ECAI), Der Haague (NL)*, pages 698–706, 2016.

[3] Mustafa Al-Bakri, Manuel Atencia, Steffen Lalande, and Marie-Christine Rousset. Inferring same-as facts from linked data: an iterative import-by-query approach. In *Proc. 29th AAAI Conference on Artificial Intelligence, Austin (TX US)*, pages 9–15. AAAI Press, 2015.

[4] Manuel Atencia, Michel Chein, Madalina Croitoru, Jérôme David, Michel Leclère, Nathalie Pernelle, Fatiha Saïs, François Scharffe, and Danai Symeonidou. Defining key semantics for the RDF datasets: experiments and evaluations. In *Proc. 21st International Conference on Conceptual Structures (ICCS), Iasi (RO)*, volume 8577 of *Lecture Notes in Computer Science*, pages 65–78. Springer, 2014.

[5] Manuel Atencia, Jérôme David, and Jérôme Euzenat. Data interlinking through robust linkkey extraction. In *Proc. 21st European Conference on Artificial Intelligence (ECAI)*, pages 15–20. IOS Press, 2014.

[6] Manuel Atencia, Jérôme David, and Jérôme Euzenat. What can FCA do for database linkkey extraction? In *Proc. 3rd ECAI workshop on What can FCA do for Artificial Intelligence? (FCA4AI), Praha (CZ)*, volume 1257 of *CEUR Workshop Proceedings*, pages 85–92. CEUR-WS.org, 2014.

[7] Manuel Atencia, Jérôme David, and François Scharffe. Keys and pseudo-keys detection for web datasets cleansing and interlinking. In *Proc. 18th international conference on knowledge engineering and knowledge management (EKAW), Galway (IE)*, volume 7605 of *Lecture Notes in Computer Science*, pages 144–153. Springer, 2012.

[8] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The description logic handbook: theory, implementations and applications*. Cambridge University Press, 2003.

[9] Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli. Characterizing functional dependencies in formal concept analysis with pattern structures. *Annals of mathematics and artificial intelligence*, 72(2):129–149, 2014.

[10] Chris Bizer, Tom Heath, and Tim Berners-Lee. Linked data — the story so far. *International Journal of Semantic Web Information Systems*, 5(3):1–22, 2009.

[11] Agnès Braud, Xavier Dolques, Marianne Huchard, and Florence Le Ber. Generalization effect of quantifiers in a classification based on relational concept analysis. *Knowledge-based systems*, 160:119–135, 2018.

[12] Dan Brickley and R.V. Guha. RDF Schema 1.1. Recommendation, W3C, 2014. https://www.w3.org/TR/rdf-schema/.

[13] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Keys for free in description logics. In *Proc. Description logic workshop (DL)*, pages 79–88, Aachen (DE), 2000.

[14] Peter Christen. *Data Matching—Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer, Heidelberg (DE), 2012.

[15] Víctor Codocedo, Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli. Characterization of order-like dependencies with formal concept analysis. In *Proc. 30th International Conference on Concept Lattices and Their Applications (CLA), Moscow (RU)*, volume 1624 of *CEUR Workshop Proceedings*, pages 123–134. CEUR-WS.org, 2016.

[16] Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 concepts and abstract syntax. Recommendation, W3C, 2014. http://www.w3.org/TR/rdf11-concepts/.

[17] János Demetrovics, Leonid Libkin, and Ilya Muchnik. Functional dependencies in relational databases: a lattice point of view. *Discrete Applied Mathematics*, 40(2):155–185, 1992.

[18] Ahmed Elmagarmid, Panagiotis Ipeirotis, and Vassilios Verykios. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, 19(1):1–16, 2007.

[19] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer, Heidelberg (DE), 2nd edition, 2013.

[20] Houssameddine Farah, Danai Symeonidou, and Konstantin Todorov. KeyRanker: Automatic RDF key ranking for data linking. In *Proc. Knowledge Capture Conference (K-CAP), Austin (TX US)*, pages 7:1–7:8, 2017.

[21] Alfio Ferrara, Andriy Nikolov, and François Scharffe. Data linking for the semantic web. *International Journal of Semantic Web and Information Systems*, 7(3):46–76, 2011.

[22] Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. In Harry S. Delugach and Gerd Stumme, editors, *International Conference on Conceptual Structures (ICCS)*, volume 2120 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2001.

[23] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: mathematical foundations*. Springer, Berlin, 1999.

[24] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.

[25] Aidan Hogan, Antoine Zimmermann, Jürgen Umbrich, Axel Polleres, and Stefan Decker. Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora. *Journal of Web Semantics*, 10:76–110, 2012.

[26] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. TANE: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111, 1999.

[27] Robert Isele and Christian Bizer. Active learning of expressive linkage rules using genetic programming. *Journal of web semantics*, 23:2–15, 2013.

[28] Mehdi Kaytoue, Sergei O. Kuznetsov, Amedeo Napoli, and Sébastien Duplessis. Mining gene expression data with pattern structures in formal concept analysis. *Information Science*, 181(10):1989–2001, 2011.

[29] Sergei Kuznetsov and Sergei Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of experimental and theoretical artificial intelligence*, 14:189–216, 2002.

[30] Mark Levene. A lattice view of functional dependencies in incomplete relations. *Acta cybernetica*, 12(2):181–207, 1995.

[31] Stéphane Lopes, Jean-Marc Petit, and Lotfi Lakhal. Functional and approximate dependency mining: database and FCA points of view. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):93–114, 2002.

[32] Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, nominals, and concrete domains. *Journal of Artificial Intelligence Research*, 23:667–726, 2005.

[33] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 Web Ontology Language: Structural specification and functional-style syntax (2nd edition). Recommendation, W3C, 2012. https://www.w3.org/TR/owl2-syntax/.

[34] Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. A survey of current link discovery frameworks. *Semantic Web*, 8(3):419–436, 2017.

[35] Axel-Cyrille Ngonga Ngomo and Sören Auer. LIMES: A time-efficient approach for large-scale link discovery on the web of data. In *Proc. 22nd International Joint Conference on Artificial Intelligence (IJCAI), Barcelona (ES)*, pages 2312–2317, Barcelona (ES), 2011.

[36] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. EAGLE: Efficient active learning of link specifications using genetic programming. In *Proc. 9th ESWC, Heraklion (GR)*, volume 7295 of *Lecture Notes in Computer Science*, pages 149–163. Springer, 2012.

[37] Eugene Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2):243–250, 1978.

[38] Nathalie Pernelle, Fatiha Saïs, and Danai Symeounidou. An automatic key discovery approach for data linking. *Journal of Web Semantics*, 23:16–30, 2013.

[39] Nikita Romashkin. FCA library, 2011. https://github.com/ae-hse/fca.

[40] Mohamed Rouane-Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. Relational concept analysis: mining concept lattices from multi-relational data. *Annals of Mathematics and Artificial Intelligence*, 67(1):81–108, 2013.

[41] Mohamed Rouane-Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. Soundness and completeness of relational concept analysis. In Peggy Cellier, Felix Distel, and Bernhard Ganter, editors, *Proc. 11h International Conference on Formal Concept Analysis (ICFCA)*, volume 7880 of *Lecture Notes in Artificial Intelligence*, pages 228–243. Springer, 2013.

[42] Fatiha Saïs, Nathalie Pernelle, and Marie-Christine Rousset. L2R: A logical method for reference reconciliation. In *Proc. 22nd National Conference on Artificial Intelligence (AAAI), Vancouver (CA)*, pages 329–334. AAAI Press, 2007.

[43] Mohamed Ahmed Sherif, Kevin Dreßler, Panayiotis Smeros, and Axel-Cyrille Ngonga Ngomo. Radon - rapid discovery of topological relations. In *Proc. 31st AAAI Conference on Artificial Intelligence, San Francisco (CA US)*, pages 175–181, 2017.

[44] Mohamed Ahmed Sherif, Axel-Cyrille Ngonga Ngomo, and Jens Lehmann. Wombat - A generalization approach for automatic link discovery. In *Proc. 14th European semantic web conference (ESWC), Portorož (SL)*, volume 10249 of *Lecture Notes in Computer Science*, pages 103–119. Springer, 2017.

[45] Yannis Sismanis, Paul Brown, Peter Haas, and Berthold Reinwald. GORDIAN: efficient and scalable discovery of composite keys. In *Proc. 32nd international conference on very large databases (VLDB)*, pages 691–702, 2006.

[46] Fabian Suchanek, Serge Abiteboul, and Pierre Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *PVLDB*, 5(3):157–168, 2012.

[47] Danai Symeonidou, Vincent Armant, Nathalie Pernelle, and Fatiha Saïs. SAKey: Scalable almost key discovery in RDF data. In *Proc. 13th International Semantic Web Conference (ISWC), Riva del Garda (IT)*, volume 8796 of *Lecture Notes in Computer Science*, pages 33–49. Springer, 2014.

[48] Danai Symeonidou, Luis Galárraga, Nathalie Pernelle, Fatiha Saïs, and Fabian M. Suchanek. VICKEY: Mining conditional keys on knowledge bases. In *Proc. 16th International Semantic Web Conference (ISWC), Wien (AT)*, volume 10587 of *Lecture Notes in Computer Science*, pages 661–677. Springer, 2017.

[49] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk – A link discovery framework for the web of data. In *Proc. WWW Workshop on Linked Data on the Web, LDOW, Madrid (SP)*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.