Data interlinking with relational concept analysis

Jérémy Vizzini

To cite this version:

Jérémy Vizzini. Data interlinking with relational concept analysis. Artificial Intelligence [cs.AI]. 2017. hal-01661184

UNIVERSITÉ
Grenoble
Alpes

Master of Science in Informatics at Grenoble
Master Informatique
Specialization Data science

Data interlinking with relational concept analysis

Jérémy Vizzini

20 June, 2017

Research project performed at LIG

Under the supervision of:
Dr. Jérôme Euzenat (INRIA)
Dr. Jérôme David (UGA)

Defended before a jury composed of:
Prof. Amini Massih-Reza (UGA)
Dr. Jean-Baptiste Durand (UGA)

June 2017

Abstract

Vast amounts of RDF data are made available on the web by various institutions providing overlapping information. To be fully exploited, different representations of the same object across various data sets have to be identified. This is what is called data interlinking. One novel way to generate such links is to use link keys. Link keys generalise database keys by applying them across two data sets. The structure of RDF makes this problem much more complex than for relational databases for several reasons. An instance can have multiple values for a given attribute. Moreover, values of properties are not necessarily datatypes but other instances of the graph. A first method has been designed to extract and select link keys from two classes of objects which deals with multiple values but not object values. Moreover, the extraction step has been rephrased in formal concept analysis (FCA) allowing to generate link keys across relational tables. Our aim is to extend this work so that it can deal with multiple values. Then, we show how to use it to deal with object values when the data set is cycle free. This encoding does not necessarily generate the optimal link keys. Hence, we use relational concept analysis (RCA), an extension of FCA taking relations between concepts into account. We show that a new expression of this problem is able to extract the optimal link keys even in the presence of circularities. Moreover, the elaborated process does not require information about the alignments of the ontologies to find out for which pairs of classes to extract link keys. We implemented these methods and evaluated them by reproducing the experiments made in previous studies. This shows that the method extracts the expected results as well as (also expected) scalability issues.

Résumé

Une grande quantité de données RDF est disponible sur le web par divers institutions créant des chevauchements d'informations. Afin d'être pleinement exploité, différentes représentations d'un même objet provenant de différents ensembles de données doivent être identifiées. C'est ce qu'on appelle le liage de données. Une nouvelle façon de générer de tels liens consiste à utiliser la notion de clés de liage. Les clés de liage génélarisent les clés en base de données en les appliquant à deux ensembles de données distincts. La structure de RDF rend ce problème beaucoup plus complexe que pour les bases de données relationnelles pour plusieurs raisons. Tout d'abord, une instance peut avoir plusieurs valeurs pour un attribut donné. De plus, les valeurs des propriétés ne sont pas forcément de types simples, ils peuvent tout aussi être d'autres instances du graphe. Une première méthode a été conçue afin d'extraire et de sélectionner des clés de liage à partir de deux classes d'objets composés par plusieurs propriétés ayant seulement des valeurs de types simples. Par ailleurs, l'étape d'extraction a été reformulée en analyse de concept formel (FCA) permettant de générer des clés de liaison pour des tables de bases de données relationnelles. Notre objectif est d'étendre ce travail afin qu'il puisse gérer de multiples valeurs. D'abord , nous montrons comment l'utiliser pour traiter les propriétés objet lorsque le jeu de données est exempt de cycles. Cet encodage ne génère pas nécessairement les clés de liage optimales. Par conséquent, nous utilisons l'analyse de concept relationnel (RCA), une extension de FCA prenant en compte les relations entre les concepts. Nous montrons qu'une nouvelle expression de ce problème est capable d'extraire les clés de liage de manière optimale même en présence de circularités. En outre, le processus élaboré ne requiert pas d'information à propos des alignements des classes des ontologies. Nous avons mis en œuvre ces méthodes et les avons évaluées en reproduisant les expériences réalisées lors d'études antérieures. Cela nous a permis de montrer que la méthode extrait les résultats attendus ainsi que de mettre en évidence un problème de mise à l'échelle (également attendus).

Acknowledgements

First, I would like to express my gratitude to my two supervisors Jérome David and Jérôme Euzenat. They bring me useful comments, remarks and engagement through this learning process. I would like to especially thank them to steer me in the right the direction during these last months. Second, I should thank the LabEx PERSYVAL-Lab for it supports. This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'avenir. Without this help, I would not be able to focus on this project. Third, I must express my very profound gratitude to my dad and to my sister for providing me with unfailing support and continuous encouragement throughout my years of study. Thank you.

Contents

Abstract i

Résumé i

1 Introduction 1

2 Problem overview: Data interlinking 3

2.1 Ontology representation . . . 3

2.2 The problem of data interlinking . . . 5

2.3 Conclusion . . . 6

3 State of the art: Link key, FCA and RCA 7

3.1 Link keys . . . 7

3.2 Formal concept analysis . . . 10

3.3 Relational concept analysis . . . 12

3.4 Conclusion . . . 16

4 Data interlinking with FCA 17

4.1 Class linking encoding . . . 17

4.2 Ontology linking process . . . 19

4.3 FCA process example . . . 20

4.4 Conclusion . . . 21

5 Data interlinking with RCA 23

5.1 Dependency problem . . . 23

5.2 RCA encoding . . . 24

5.3 RCA process example . . . 25

5.4 Concepts and link keys . . . 30

5.5 Noisy example . . . 31

5.6 Conclusion . . . 33

6 Implementation and experiments 35

6.1 Implementation . . . 35

6.2 Experiments . . . 36

6.3 Conclusion . . . 39

7 Conclusion **41**

Bibliography **43**

— 1 —
Introduction

Context

Vast amounts of data are made available in RDF (Resource description framework) on the web. RDF representations use annotated graphs to model information as well as the relationships between each element composing it. The annotations are usually specified in an ontology defining classes to which resources belong and properties between them. Data sets are made available by various institutions providing overlapping information.

Problem

To be fully exploited, different representations of the same object in various data sets have to be identified. This is what is called data interlinking. To illustrate the problem, we can think about two social networks belonging to a unique company. As one can imagine, a company has benefits into combining the information of its two networks in order to sell more valuable advertisements. In other words, the firm has an interest in identifying same users of the two networks in order to determine their profile in a more accurate manner and by consequence sell more expensive advertisements justifying that they now fit better to each user profile. Here, data interlinking amounts at identifying the same users across data sets. For that purpose, it needs to determine the information that allows us to identify uniquely individual.

Approach

The literature already contains multiple methods performing data interlinking. One novel way to generate such links is to use link keys [2]. Link keys generalise database keys by applying them across two data sets. In the example, the problem of link keys extraction can be reformulated as follows: What are the (minimal) sets of properties allowing to identify resources from two data sets? Are the name and the birthday enough information to identify individual or should we use other elements ? The structure of RDF makes this problem much more complex than for relational databases for several reasons. In RDF, an instance can have multiple values for a given attribute. Moreover, values of properties are not necessarily datatypes. They can also be other instances of the graph. Hence, in such a situation, the equality of two properties have to be defined according to the data. For instance, in order to link books, a possible link key would be composed by two conditions on the title attribute and the author attribute. Books would be identified by the fact that they share the same title as well as at least one author. Similarly a second link key would be used to link authors. For example, by the fact that they have the same name and the same parents. This notion allows both linking data with precision and obtaining human interpretable linking function.

A first method has been designed to extract link keys from two classes of objects which deals with multiple values but not object values [2]. This method works in two steps: extraction of candidate link key and selection. The extraction step has been rephrased in formal concept analysis (FCA) allowing to generate link keys across relational tables [3].

Contribution

Our aim is to extend this work by addressing the above mentioned problems (multiple values and object values). For that purpose, we first extend the proposed FCA encoding so that it can deal with multiple values. Then, we show how to use it to deal with object values when the data set is cycle free. This encoding does not necessarily generate the optimal link keys, as selected by the selection functions of [2].

Hence, we use relational concept analysis (RCA), an extension of FCA taking relations between concepts into account. We show that a new expression of this problem is able to extract the optimal link keys even in the presence of circularities. Moreover, the elaborated process does not require information about the alignments of the ontologies to find out for which pairs of classes to extract link keys. We implemented these methods and evaluated them by reproducing the experiments made in [2]. This shows that the method extracts the expected results as well as (also expected) scalability issues.

Outline of the report

This dissertation is composed of five chapters. Chapter 2 defines the problem. We especially describe what is an ontology. Then, the domain of ontology matching and data interlinking are presented. Chapter 3 introduces the fundamental notions used in this document. We especially define the notion of link key and candidate link key. We describe precisely and also illustrate the two notions. Formal concept analysis is then presented. An example illustrates the notions and then relational concept analysis is introduced. Chapter 4 presents the FCA encoding of the link key extraction problem for two data sets where the classes are aligned. Chapter 5 specifies our RCA encoding. Chapter 6 describes our implementation and presents experiments evaluating the proposed approach.

— 2 —
Problem overview: Data interlinking

The first section of this chapter briefly presents ontologies. Instead of formalising the notion, we start by suggesting an informal description of this representation. We then expose how to express them thanks to RDFS and OWL. The second section of this chapter introduces the problem of data interlinking. The difference between the two domains ontology matching and data interlinking is presented. It follows a presentation of the different approaches.

2.1 Ontology representation

Data sets are expressed in RDF. The vocabulary offered by such data set may be expressed in an ontology. One fundamental advantage of this representation is that ontologies can be interpreted and processed by machines thanks to a logical semantics that enables reasoning.

URI and RDF

Resource description framework is a standardised way to express data [5]. RDF data are made of elements and relationships connecting them. These elements are called resources. A resource is anything that can be referred to an element composing the information. It can be a Web page, an XML document, a Web service, an identifier for an entity or even a concept. Resources are identified by URIs, short for Uniform Resource Identifiers (noted as $\mathscr{U}$). An URI may notably be a URL that any (human or software) agent or application can access.

RDF uses the structure subject–predicate–object to model relationship between two resources. A RDF triple formally belongs to $\{\mathscr{U} \cup \mathrm{BN}\} \times \mathscr{U} \times \{\mathscr{U} \cup \mathrm{LT} \cup \mathrm{BN}\}$ where LT denotes literal elements such as primitive values and BN denotes the blank nodes. The subject corresponds to a resource, the predicate denotes some aspects of this resource, and the object refers to the element in relation with the subject according to the predicate.

Figure 2.1 is an example composed of elements about the film Titanic. The schema is composed of resources identified by URLs. Thanks to RDF, we can especially model that L. DiCaprio is an actor of the movie Titanic. In our case, the subject is the concept of ‘DiCaprio’ identified by `http://www.movie.web/dicaprio/`, the object is the movie ‘Titanic’ identified by `http://www.movie.web/titanic/` and the predicate is the concept of ‘acting’ identified by `http://www.movie.web/actor/actedin`. This triple composes a edge of the ontology graph and it models the relationship.

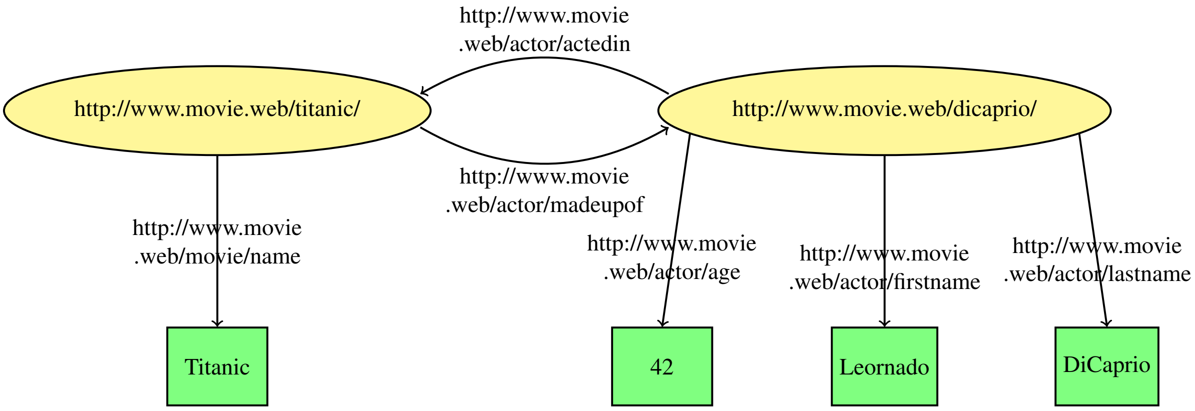


Figure 2.1 – RDF graph about the film Titanic

RDFS and OWL

The RDF standard does not provide any specification about the semantic of the terms, i.e., URIs, used. Ontologies do that. For that purpose, they provide a reserved set of URIs (prefixed by RDFS or OWL). This vocabulary allows to specify the meaning of the terms according to a logical semantics.

More precisely, ontologies can be defined as follows:

Definition 1 (Ontology). An ontology $\langle C, P, A\rangle$ is made of a set of concepts $C \subseteq \mathscr{U}$, a set of properties $P \subseteq \mathscr{U}$, $C \cap P = \emptyset$ and A a set of axioms with concepts C and role have P in a description logic. [8]

Definition 2 (Entailment). The entailment $\models$ is a relation between an ontology $\mathscr{O} = \langle C, P, A\rangle$ and a RDF triple $\langle s, p, o\rangle$ such that $p \in P$. To simplify the notation, we denote by $p(o) = \{v \mid \mathscr{O} \models \langle o, p, v\rangle\}$ when $\mathscr{O}$ is implicit.

RDFS, short for Resource Description Framework Schema ([13]) provides a standard way to constain relationship between classes and properties. For that purpose, it provides terms such as `rdfs:subClassOf`, `rdfs:subPropertiesOf`, `rdfs:domain` and `rdfs:range`. Figure 2.2 contains a RDFS graph about the film Titanic illustrating these terms.

The Web Ontology Language (OWL) [9] has been introduced to enforce the notion of structure of classes. Analogous to RDFS, it provides additional terms to specify resource structure. It makes more expressive the vocabulary and by the way more complex the reasoning. Ìn Figure 2.2, it provides terms such as `owl:inverseOf` indicating that the property `movie:made_by` is the reciprocal property of `movie:made`.

The OWL framework also provides new terms in order to deal with duplicates. In the rest of this work, we will use two of them `owl:equivalentClass` and `owl:sameAs`. They can be used to respectively express equality statements between two classes and two instances.

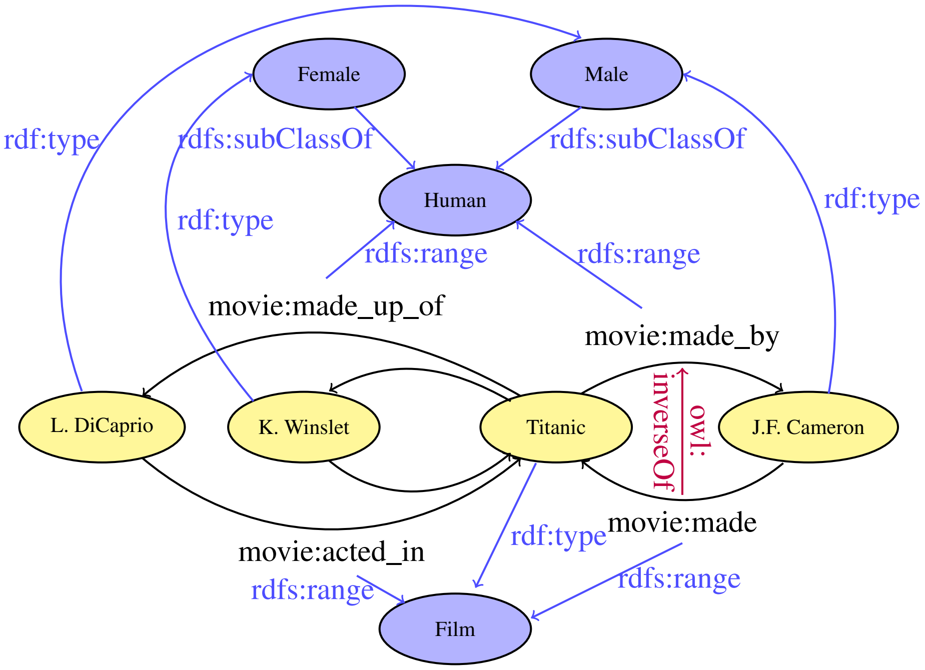


Figure 2.2 – RDF, RDFS and OWL representation of the film Titanic

2.2 The problem of data interlinking

The initiative instituted by the semantic web community of elaborating the web of data has encouraged the proliferation of thousands data sets through the web. Due to its decentralised aspect and its scalability, many ontologies have emerged for every specific domain. Moreover some of them did appear for same domains creating duplicates.

Ontology matching is the task of identifying correspondences between the schemata structuring the data [6]. This is what is called an alignment. It corresponds into finding classes as well as properties specifying the same meaning. However, such information is not sufficient and new algorithms have to be developed especially for the task of identifying similar instances.

Data interlinking

Data interlinking refers to the task of finding same instances present in different RDF data sets. More formally, from two RDF data sets d and d', we call data interlinking the process of generating `owl:sameAs` triples identifying pairs of same resources coming from d and d'.

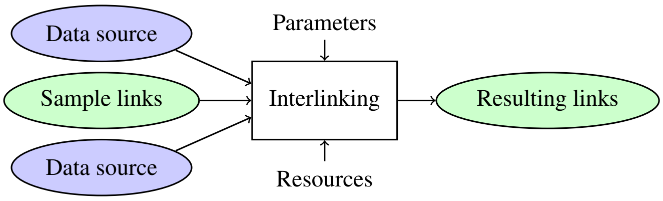


Figure 2.3 – Data interlinking process

Data interlinking approaches

Many data interlinking methods and approaches have been elaborated through the years. [17] and [14] make a review of many data interlinking systems. The literature contains two main approaches to data interlinking: one based on similarity measures and a second one based on the extraction of keys.

Similarity-based methods compare resources through a similarity measure. If two instances are similar enough, then they are considered as the same and the relative `sameAs` link is generated. The similarity is either based on the values of the instances or based on the graph structure. Values-similarity-based techniques use normalisation and translation techniques along with information retrieval measures such as TF-IDF. Graph-similarity-based methods reuse graph matching techniques.

Key-based methods aims at determining sufficient conditions for two resources to be the same. If two instances satisfy the conditions, then they are considered as the same and the relative `sameAs` link is generated. Our approach belongs to one of them by using as conditions link keys. This approach is further detailed in the next chapter.

2.3 Conclusion

In this chapter, we presented what are RDF data sets and ontologies and how to express them thanks standard languages. We then presented the data interlinking problem as well as the different approach to resolve it.

Our goal is to address it using the link key approach. Hence, the next chapter will provide further details on link keys. We will also explain formal concept analysis as well as relational concept analysis. that we use.

— 3 —

State of the art: Link key, FCA and RCA

This chapter presents state of the art techniques used in this work. We start by defining the notion of candidate link key and how to select the best ones. Then formal concept analysis is introduced. From this point, we initiate readers to its extension named relational concept analysis which is widely used by our second encoding proposal.

3.1 Link keys

A link key is a set of pairs of properties generalising the idea of aligned keys. A pair of aligned keys corresponds to properties from both ontologies allowing to link instances. Link keys are defined with respect to the equality statement between the pairs of values for each pair of properties.

Definition 3 (**Link key**). Given two ontologies $\mathscr{O} = \langle C,P,A \rangle$ and $\mathscr{O}' = \langle C',P',A' \rangle$, two aligned classes $(c,c') \in C \times C'$, with p, p' and q, q' respectively properties of the classes c and c', a link key is

$$\langle \{\langle p_1,q_1 \rangle,...,\langle p_k,q_k \rangle\}\{\langle p'_1,q'_1 \rangle,...,\langle p'_l,q'_l \rangle\}\{\langle p''_1,q''_1 \rangle,...,\langle p''_l,q''_m \rangle\} \text{ link key } \langle c,c' \rangle \rangle$$

holds if and only if $\forall o; \mathscr{O} \models c(o), \forall o'; \mathscr{O}' \models c'(o')$, we have :

$$\begin{aligned} \text{if } \forall i \in 1,...,k : p_i(o) \cap q_i(o') \neq \emptyset \quad (3.1)\\ \text{and } \forall i \in 1,...,l : p'_i(o) = q'_i(o') \quad (3.2)\\ \text{and } \forall i \in 1,...,m : p''_i(o) = q''_i(o') \neq \emptyset \quad (3.3)\\ \text{then } \langle o, \text{owl:sameAs}, o' \rangle \text{ holds} \end{aligned}$$

The meaning of the three conditions 3.1, 3.2 and 3.3 corresponds respectively to the in-link key, the eq-link key and the in-eq-link key that we present below. We illustrate the definition via two examples about two ontologies `myLab` and `myCity`. These data sets have two equivalent classes `myLab:Researcher` and `myCity:Inhabitant` both representing human being.

in-link key condition

In Figure 3.1, the instances `myLab:TomasDupond` and `myCity:TomDupond` represent the same person but in the two different ontologies. We can notice that they do not share exactly all their hasEmail values but at least they share one. Hence, in this kind of situation, expecting an equality is a to strong assumption and most instances would not be linked. However sharing at least one value is sufficient. This is what is done by the properties p_i and q_i in the condition 3.1. The link key identifies a correspondence if and only if at least one value is shared by the instances for pairs of given properties.

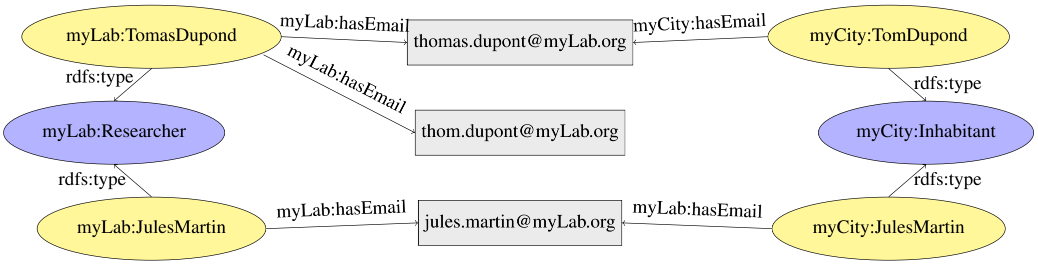


Figure 3.1 – A RDF graph illustrating the in-condition of a link key

eq-link key condition

The second situation represented by Figure 3.2 is exactly the opposite of the first one. The instances of the RDF graph are this time linked to research articles. If we try to link instances sharing at least one property, too many correspondences are found. Moreover some of them are wrong. Hence, the first condition is too ambiguous. That is why a link key is also composed of pairs of properties p'_i and q'_i which establish connections between the two classes if for the given properties all the values are shared. A complete equality is required to reveal the expected equivalence.

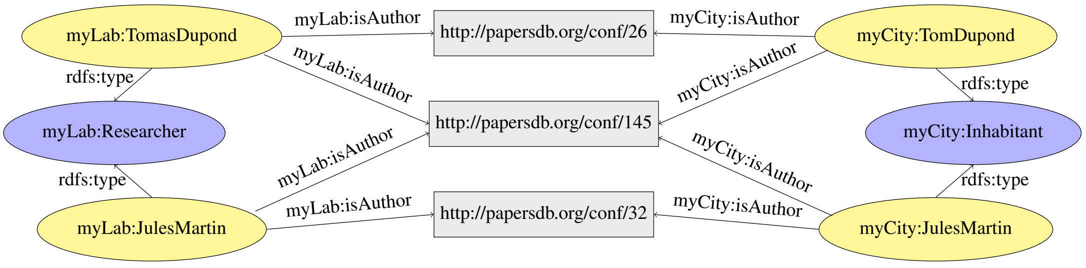


Figure 3.2 – A RDF graph illustrating the eq-condition of a link key

The difference between the condition 3.2 and 3.3 is simply that the condition 3.2 links instances if each instance has no value for the given pairs of properties whereas the condition 3.3 requires the existence of at least one value for each one. One can notice that the condition 3.3 entails both conditions 3.1 and 3.2. We denote the condition 3.3 by the term in-eq-link key. An evaluation of the two approaches (eq and in) can found in [1].

As the equality of values from the link key conditions can be too restrictive, a first step is to use methods such as value clustering or normalization to partially solve the problem.

Selection of link keys

[2] has proposed an algorithm allowing to extract link keys. The method operates in two steps. It firstly extract a set of candidate link keys. A candidate link key is a set of pairs of properties that

can generate at least one link if it is used as a link key. The second step is to select some of the extracted candidate link keys as link keys. In order to make this selection, measures have been proposed by the authors. The article suggests two approaches: a supervised and an unsupervised selection. We present below these two methods.

Supervised selection measures

The supervised approach consists of using the classical precision and recall measures on a part of the data that has been already linked by a reference. We denote by L^+ positive examples annotated by `owl:sameAs` links. Similarly L^- denotes the negative examples annotated by `owl:differentFrom` links.

Definition 4 (Precision and recall). The precision measure is the fraction of retrieved links that are relevant to the ground truth. The recall measure is the fraction of the reference links that are successfully retrieved.

We denote by k a candidate link key and by $L_{D,D'}(k)$ the generated links between D and D'.

$$\text{precision}(k) = \frac{\mid L^+ \cap L_{D,D'}(k) \mid}{\mid (L^+ \cup L^-) \cap L_{D,D'}(k) \mid}$$

$$\text{recall}(k) = \frac{\mid L^+ \cap L_{D,D'}(k) \mid}{\mid L^+ \mid}$$

As mentioned by the authors, given a $L^- = \emptyset$, the precision would not be relevant. A possible workaround is to artificially generate `owl:differentFrom` links according to the known `owl:sameAs` links. The idea is to close the `owl:sameAs` links by adding to L^- links `owl:differentFrom` for every other instances that does not hold `owl:sameAs` links. In this way, L^- is not empty anymore and the f-measure is simply calculated by

$$\text{f-measure}(k) = \frac{2 * \text{precision}(k) * \text{recall}(k)}{\text{precision}(k) + \text{recall}(k)}$$

Unsupervised selection measures

Most of the time, a linked data set of reference is not available. In this kind of situation, we can only make measures based on local instances by estimating the correctness of potentially generated links according to a given link key. For that purpose, two measures has been introduced: the discriminability and the coverage. The proposal is to measure how close the selected link key is able to generate a one-to-one mapping for each linked instances. This is what is evaluated by the discriminability measure. Furthermore, a second proposal is that a link key generating a lot of links is more interesting than one generating few links. This is what is calculated by the coverage measure. It evaluates the proportion of instances of both classes that are linked by a given candidate link key.

Definition 5 (Discriminability and coverage). The discriminability estimates the closeness to a one-to-one mapping. The coverage evaluates the proportion of linked instances compared to the full set:

$$\text{discriminability}(k) = \frac{min(\mid \{a : \langle a,b \rangle \in L_{D,D'}(k)\} \mid, \mid b : \langle a,b \rangle \in L_{D,D'}(k) \mid \})}{\mid L_{D,D'}(k) \mid}$$

$$\text{coverage}(k) = \frac{\mid \{a : \langle a,b \rangle \in L_{D,D'}(k)\} \cup \{b : \langle a,b \rangle \in L_{D,D'}(k)\} \mid}{\mid \{a : c(a) \in D\} \cup \{b : d(b) \in D'\} \mid}$$

The coverage measure tends to select less specific link key in order engender more links. On the opposite, the discriminability tends to select very specific ones in order to engender a perfect one-to-one mapping. Similarly to the use of a f-measure, the harmonic mean of these two values allows to obtain both characteristics.

3.2 Formal concept analysis

Formal concept analysis provides techniques to extract concepts between two interdependent sets. [7] is the reference of the domain, the definitions used in the rest of this section comes from it. FCA is particularly attractive because of its closeness to the notion of classes also used in the ontology model. Moreover, this domain was studied for years and provides advanced mathematical tools because of its foundation based on Galois connections.

Concepts and contexts

FCA introduces the notion of formal context. It is an incidence relation between objects and attributes. It is from where concepts are extracted. The next definition describes formally the idea.

Definition 6 (Formal context). A formal context $\mathbb{K} := (G,M,I)$ consists of two sets G and M and a relation I between G and M. The elements of G are called the objects and the elements of M are called the attributes of the context. In order to express that an object g is in relation I with an attribute m, we write gIm or $(g,m) \in I$ and read it as 'the object g has the attribute m'.

Figure 3.3 is an example of binary tables representing a formal context. This context is defined by its attributes (activities) and its objects (living being). The presence of relation is denoted by a 1 in the table otherwise the absence is denoted by a 0. In the rest of this document, on the opposite of the usual representation of the domain, the incidence table has its extent horizontally and its intent vertically. This choice makes the table smaller and the document easier to read.

	$g_1 =$ Kids	$g_2 =$ Adults	$g_3 =$ Dogs
$m_1 =$ Go to work	0	1	0
$m_2 =$ Go to school	1	0	0
$m_3 =$ Go to the vet	0	0	1
$m_4 =$ Eat food	1	1	1
$m_5 =$ Drive cars	0	1	0
$m_6 =$ Do sport	1	1	0

Figure 3.3 – Table representing a FCA context

From such a context, FCA aims at extracting concepts. In order to be able to define concepts, we now define two concept-forming operators which characterise them.

Definition 7 (Concept-forming operators). For a formal context $\langle G, M, I \rangle$, operators $\uparrow : 2^G \rightarrow 2^M$ and $\downarrow : 2^M \rightarrow 2^G$ are defined for every $A \subseteq G$ and $B \subseteq M$ by

$$A^{\uparrow} = \{ \, m \in M \;\mid\; \forall g \in A : \langle g, m \rangle \in I \, \},$$
$$B^{\downarrow} = \{ \, g \in G \;\mid\; \forall m \in B : \langle g, m \rangle \in I \, \}.$$

In simple words, the $\uparrow$ operator collects the objects in relation with all the given attributes. In the same way, the $\downarrow$ operator collects the attributes in relation with all the given objects. For instance, in the example of the Figure 3.3, we can extract the following results:

$$\{m1\}^{\downarrow} = \{g2\},$$
$$\{m4, m6\}^{\downarrow} = \{g1, g2\},$$
$$\{g1\}^{\uparrow} = \{m2, m4, m6\},$$
$$\{g1, g2\}^{\uparrow} = \{m4, m6\}.$$

We now define the notion of formal concept. A formal concept can be seen as the maximal set of objects sharing a maximal set of same attributes. Some concepts satisfying a lot attributes will be very specific to few objects and some others with few attributes very common to all the objects of the context.

Definition 8 (Formal concept). A formal concept of the context (G, M, I) is a pair (A, B) with $A \subseteq G, B \subseteq M$, $A^{\uparrow} = B$, and $B^{\downarrow} = A$. We call A the extent and B the intent of the concept (A, B). $\mathfrak{B}(G, M, I)$ denotes the set of all concepts of the context (G, M, I).

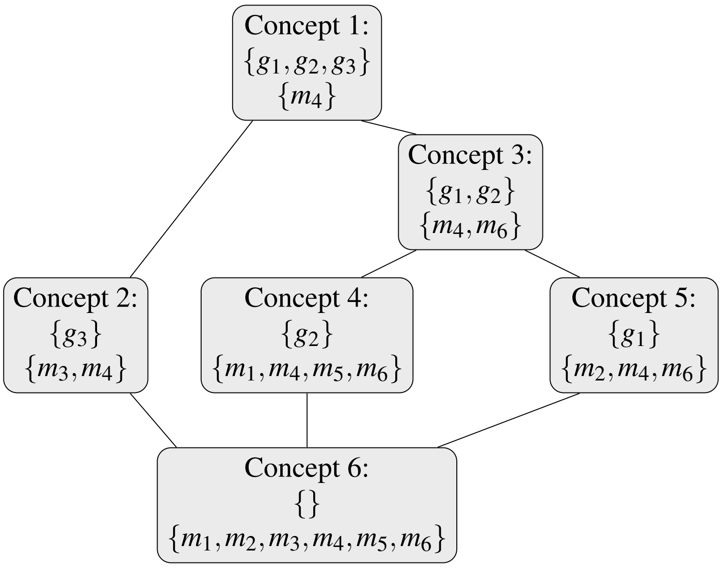
From a mathematical point of views, the formal concept (A, B) is a fixed point of the pair of operators $\langle \uparrow, \downarrow \rangle$. In the Galois theory, we say that $\langle \uparrow\downarrow, \downarrow\uparrow \rangle$ is a closure operation. The list of all the formal concepts of the Figure 3.3 are listed below. To each extent, an abstraction can be inferred. For instance, the concept C_3 gathers the objects $g_1 =$ Kids and $g_2 =$ Adults. We can identify by its extent the notion of human being.

$$C_1 = (\{g_1, g_2, g_3\}, \{m_4\}),$$
$$C_2 = (\{g_3\}, \{m_3, m_4\}),$$
$$C_3 = (\{g_1, g_2\}, \{m_4, m_6\}),$$
$$C_4 = (\{g_2\}, \{m_1, m_4, m_5, m_6\}),$$
$$C_5 = (\{g_1\}, \{m_2, m_4, m_6\}),$$
$$C_6 = (\{\}, \{m_1, m_2, m_3, m_4, m_5, m_6\}).$$

Multiple optimized algorithms have been elaborated to extract concepts and lattices. We did not present a particular FCA algorithm here. [12] makes a review of some of them. The choice of the method depends on the applications.

Concept lattices

The concepts extracted in the previous section can be partially ordered by inclusion. A formal concept is composed of two sets: an extent and an intent. In this way, it is possible to order the concepts by using the inclusion order of one these sets. In the case of an ordering on the attributes, we denote by $C_i \leq C_j$ where the concept $C_i = (G_i, M_i)$ and the concept $C_j = (G_j, M_j)$ when we have $M_i \supseteq M_j$. The partially order on the objects is defined similarly whenever we have $G_i \subseteq G_j$. The partially order of concepts is dual. It means that if we have $M_i \subseteq M_j$ as ordering function, it implies that $G_i \supseteq G_j$ and reciprocally. From this partial ordering, a complete lattice can be build as follows :



Concept 1: $\{g_1, g_2, g_3\}$ $\{m_4\}$
Concept 3: $\{g_1, g_2\}$ $\{m_4, m_6\}$
Concept 2: $\{g_3\}$ $\{m_3, m_4\}$
Concept 4: $\{g_2\}$ $\{m_1, m_4, m_5, m_6\}$
Concept 5: $\{g_1\}$ $\{m_2, m_4, m_6\}$
Concept 6: $\{\}$ $\{m_1, m_2, m_3, m_4, m_5, m_6\}$

Figure 3.4 – Complete lattice from a context

One can notice that the higher the node, the less specific the concept. For instance, concept 1 is at the top of the lattice and covers the three objects. This concept is very general. It refers to the ‘living being’ concept. If we go down in the lattice, we obtain more specific concepts. Concept 2 generalizes the notion of ‘pet’ and concept 3 the notion of ‘human’.

3.3 Relational concept analysis

Relational concept analysis which has been introduced in [10] is a way to go beyond the limitation of boolean predicates and concept descriptions by defining how to deal with intra-concepts as well as relational ones. Indeed RCA which is built on the top of FCA is able to handle circular descriptions and can be used to determine concepts embedded in relations. In the rest of this section, we mainly use definitions from [11]. We start by defining how to encode a relation in RCA. Then, we present how to scale FCA contexts in order to extract relational concepts.

Encoding relations

Similarly to the notion of formal context, RCA provides the notion of relational context and relational context family. The representations of the relations is composed as before by binary relations between some objects and some attributes. These objects and attributes are those from the formal contexts in relations.

Definition 9 (Relational context family). A Relational context family (RCF) is a pair (K,R) where K is a set of formal contexts $K_i = (G_i,M_i,I_i)$ and R is a set of relational contexts (G_k,G_i,I_j). (G_k,G_l) are the object sets of formal contexts (K_k,K_l) and $I_j \subseteq G_k \times G_l$. K_k is called the source context and K_l is called the target context.

We illustrate this definition by exhibiting an example of relational context family. This example refers to students belonging to courses. Students are characterised by their skill in different subjects. On the other hand, courses are only identifiable by the number of students. As we know which students belong to which courses, an idea is to characterises and identify the courses according to the students skills. This kind of propagation of attributes is not possible in FCA. It is the use of the relational information that determines the taught subject of each course and by the way characterise them. In the context of RCA it can be simply modelled. Each element has its own formal context and the membership information can be encoded via a relational context.

We provide below the relational context family of this example. We denote by $K_0 = (G_0,M_0,I_0)$ the formal context of the students, by $K_1 = (G_1,M_1,I_1)$ the formal context of the courses and by $R_0 = (G_1,G_0,I_2)$ the relational context between classes and students.

$I_0 \in K_0$	$g_1 =$ Student 1	$g_2 =$ Student 2	$g_3 =$ Student 3	$g_4 =$ Student 4
$m_1 =$ Mathematics	1	1	0	1
$m_2 =$ Computer	1	0	0	0
$m_3 =$ Language	0	0	1	0
$m_4 =$ Physics	0	1	0	1
$m_5 =$ Writing	0	1	1	0

$I_1 \in K_1$	$g_5 =$ Course 1	$g_6 =$ Course 2	$g_7 =$ Course 3
$m_6 =$ 20 students	1	0	0
$m_7 =$ 26 students	0	1	0
$m_8 =$ 18 students	0	0	1

I_2	g_5	g_6	g_7
g_1	1	0	0
g_2	0	1	0
g_3	0	0	1
g_4	0	1	0

Table 3.1 – Relational context family of students and courses

Scaling relations

The aim of RCA is to extract intra-concepts as well as relational concepts. For that purpose, RCA provides a tool named scaling operator. This operator has the aim of introducing new concepts in formal contexts from the relational contexts linking them.

Definition 10 (Scaling operator). A scaling operator is a function taking as input a relational context $R_j = (G_k,G_l,I_j)$, the concept set A of a lattice $\mathscr{L}_l^n$ built on objects of G_l and a pair object-concept $(g,a) \in (G_k \times A)$. This function returns true if and only if there is a relation between g and a.

Table 3.2 proposes the most used scaling operators. They corresponds to the most common descriptors from description logics [4]. The choice of the scaling operator has a direct meaning. In our example, we aim at identifying courses according to all the students belonging to them.

That is why in this situation we choose the universal operator. All the students are important to understand the course subject. We can imagine another situation such that the relation is not related to all the students but only to some of them, for instance only with the delegate students. This time, the existential operator is more suitable. Indeed, in this context, only one student is enough information to characterise uniquely a course.

Operator	Attribute form	Condition
Universal (wide)	$\forall r : c$	$r(g) \subseteq \text{Ext}(c)$
Includes	$\supseteq r : c$	$r(g) \supseteq \text{Ext}(c)$
Existential	$\exists r : c$	$r(o) \cap \text{Ext}(c) \neq \emptyset$
Universal strict	$\forall\exists r : c$	$r(g) \subseteq \text{Ext}(c)$ and $r(g) \neq \emptyset$
Qualified cardinality restriction	$\geq n, r : c$	$r(g) \subseteq \text{Ext}(c)$ and $\lvert r(g) \rvert \geq n$
Cardinality restriction	$\geq n, r : \top_{\mathscr{L}}$	$\lvert r(g) \rvert \geq n$

Table 3.2 – Most used scaling operators

Definition 11 (Scaled context). For each relational context $R_j = (G_k, G_l, I_j)$ and a scaling operator S, a scaled context $R_j^* = (G_k, A, I_j)$ where G_k the object sets of formal contexts (K_k), A is the concept set of the lattice $\mathscr{L}_l^n$ built on objects of G_l and I_j^* contains the relations $(g, a) \in (G_k \times A)$ if and only if $S(R_j(g), Ext(a))$ is true.

A scaled context is simply a formal context having as intent the concepts of the lattices built from the contexts holding a relation. We build Table 3.3, the scaled context of $R_0^* = (G_1, A, I_1^*)$. R_0^* is the scaled context of the relational context R_0. A is the concept set of the lattice built on objects of G_0, in our case the lattice built from K_0. Its concepts are listed below.

$$\begin{array}{llll}
C_1 & = (\{\}, M) & C_2 & = (\{g_1\}, \{m_1, m_2\}) \\
C_3 & = (\{g_2\}, \{m_1, m_4, m_5\}) & C_4 & = (\{g_3\}, \{m_3, m_5\}) \\
C_5 & = (\{g_2, g_3\}, \{m_5\}) & C_6 & = (\{g_2, g_4\}, \{m_1, m_4\}) \\
C_7 & = (\{g_1, g_2, g_4\}, \{m_1\}) & C_8 & = (G, \{\})
\end{array}$$

As we can see, the concept C_2 represents students studying computer science, C_4 those studying language and C_6 those studying physics. In consequence, we deduce from the scaled context that the class g_5 only have computer science students. Similarly, classes g_6 and g_7 have respectively only language and physics students. As expected, the concepts from the student contexts allows to identify the courses. The next subsection proposes an algorithm performing this identification process for us.

	g_5	g_6	g_7
C_1	0	0	0
C_2	1	0	0
C_3	0	0	0
C_4	0	0	1
C_5	0	0	1
C_6	0	1	0
C_7	1	1	0
C_8	1	1	1

Table 3.3 – Scaled context R_0^*

RCA concepts and algorithm

As demonstrated in the example, the use of a scaled context allows, to include to the initial formal context, concepts coming from other contexts having relations. The purpose of RCA is to handle relational concepts automatically in the same way as intra-concepts. Similarly to formal concepts, we define RCA concepts. The propagated concepts have only been added to the formal concept definition.

Definition 12 (Relational concept). For a relational context family $(\{K_i\}_{i=0}^n, \{R_j\}_{j=0}^m)$, a relational concept of the formal context (G,M,I) is a pair (A,B) where $A \subseteq G$ and $B = B_I \cup B_R$ with $B_I \subseteq M$ and B_R is a set of concepts such that $A^{\uparrow} = B_I$, $B_I^{\downarrow} = A$ and $c \in B_R \Leftrightarrow \exists (K_p, R_p)\ \forall g \in A$ we have $S(R_p(g), Ext(c))$ is true.

In order to extract them from a relational context family, an algorithm is provided. The method is to create at each step the scaled context of each relational context by using the lattice obtained in the previous iteration. In this way, at each iteration the process concatenates these scaled contexts to the initial ones. By repeating it multiple times, the procedure creates new concepts and propagates them. Finally the algorithm converges to a fixed point [10].

The complete method is summed up in Algorithm 1. One benefit of this algorithm is that it terminates even if the relational context family holds circular dependencies. For the moment, the convergence of the algorithm has not been proven. However, multiples arguments lets the authors think that it is the case. One argument is that the number of objects in extended contexts is not changing during the process. Hence, the number of concepts for every lattice L_i is limited to $2^{|G_i|}$.

Algorithm 1 RCA algorithm

```
INPUT:
- A relational context family $\langle \{K_i\}_{i=1}^n, \{R_j\}_{j=1}^m \rangle$ and a scaling operator $S$.
OUTPUT:
- A formal lattice for all the contexts composed of the extracted formal and relational concepts.
INIT:
- The concept lattices $\mathscr{L}_0 = \{\text{fca}(K_i)\}_{i=1}^n$
STEP $p$:
  for each formal context $K_i$ do
    - elaboration of the scaled context $R_i^* = \text{scale}(K_i, \mathscr{L}_{p-1}, \{R_j\}_{j=1}^m)$.
    - concatenation of the formal context $K_i' = \text{concatenate}(K_i, R_i^*)$.
  end for
  - update of concept lattices $\mathscr{L}_p = \{\text{fca}(K_i')\}_{i=1}^n$.
  - if $\mathscr{L}_p$ and $\mathscr{L}_{p-1}$ are isomorphic halt else increment $p$ and repeat
```

We performed the algorithm on the presented RCF example. The algorithm converged to a fixed point after two iterations and produced the two resulting lattices displayed below. As expected, the concepts characterising the students have been propagated to the courses lattice. Courses are now identifiable by other elements than the number of students. We can use the concepts coming from K_0 that are not shared by the other courses. The course g_5 is identifiable by the concept C_2 = computer science student, g_6 by C_6 = physics student and g_7 by C_4 = language student.

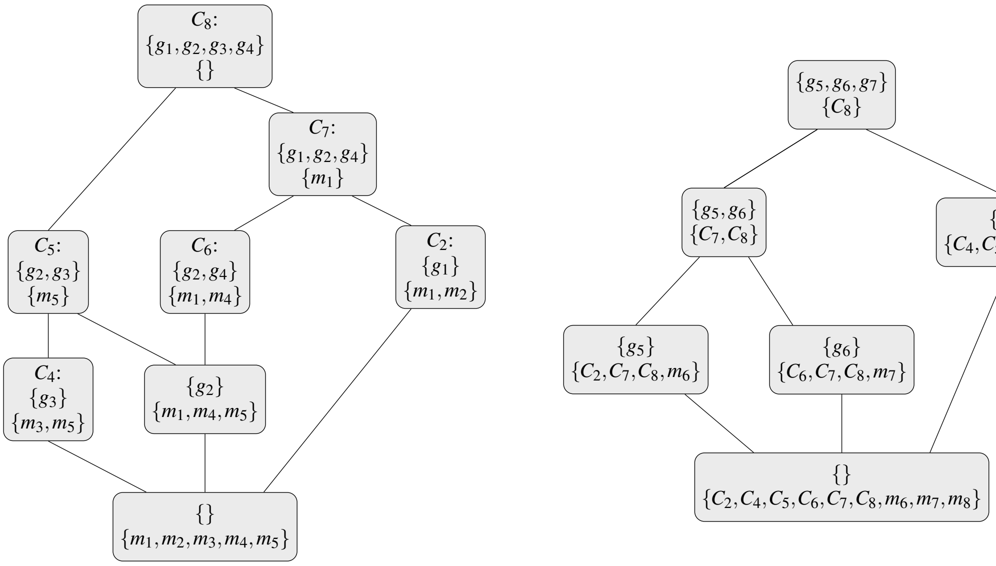

C_8: $\{g_1,g_2,g_3,g_4\}$ $\{\}$
C_7: $\{g_1,g_2,g_4\}$ $\{m_1\}$
C_5: $\{g_2,g_3\}$ $\{m_5\}$
C_6: $\{g_2,g_4\}$ $\{m_1,m_4\}$
C_2: $\{g_1\}$ $\{m_1,m_2\}$
C_4: $\{g_3\}$ $\{m_3,m_5\}$
$\{g_2\}$ $\{m_1,m_4,m_5\}$
$\{\}$ $\{m_1,m_2,m_3,m_4,m_5\}$
$\{g_5,g_6,g_7\}$ $\{C_8\}$
$\{g_5,g_6\}$ $\{C_7,C_8\}$
$\{g_7\}$ $\{C_4,C_5,C_8,m_8\}$
$\{g_5\}$ $\{C_2,C_7,C_8,m_6\}$
$\{g_6\}$ $\{C_6,C_7,C_8,m_7\}$
$\{\}$ $\{C_2,C_4,C_5,C_6,C_7,C_8,m_6,m_7,m_8\}$

Figure 3.5 – Lattice of K_0 on the left and lattice of K_1 on the right

3.4 Conclusion

In this chapter, we provided the foundation of our work. On the first hand, we saw the notion of link key. We saw how the notion of in-link key and eq-link key allow to deal with the multiplicity of the values. On the other hand, FCA and RCA have been explained. We looked at what is a formal context and what are formal concepts. We also saw how to build lattices. Then, RCA has been introduced and the notion of relational context family defined along with the RCA algorithm.

The next chapter presents our first contribution. Our aim is to extend the FCA encoding elaborated for relational tables to link two ontologies. For that purpose we consider that we know pairs of aligned classes and proposes an encoding that encodes the link key extraction problem.

— 4 —
Data interlinking with FCA

In this chapter, we propose a method using FCA to extract link keys. In [3] the link key problem has been encoded for databases thanks to the elaboration of contexts composed of pairs of attributes and pairs of tuples. Hence, the concepts resulting from this encoding have as intent a candidate link key and as extent the relative pairs of linked tuples.

Here, we adapt this method to ontologies by understanding how to deal with the EQ and IN conditions presented in the candidate link key definition. These conditions allow us to deal with the multiplicity of values. For that purpose, we assume that the alignment between the classes of the two ontologies is known. We start by explaining how to find link keys for aligned pairs of classes.

Then the final procedure is presented. In order to take charge of object properties, it consists in linking pairs of classes in a cascading way: starting from classes with no dependency as a first step to then handle classes having as dependency the classes interlinked in the previous iteration.

4.1 Class linking encoding

Our aim is to extract link keys for pairs of classes. Hence, we need to encode the link key extraction problem for a pair of aligned classes in a formal context. The idea is to build a context putting in relation every pair of instances of each class to every pair of properties quantified. Concepts resulting from the process have an extent composed of set of pairs of quantified properties constituting a candidate link key. The quantification corresponds to both EQ and IN link key. The intent are composed of the pairs of instances linked by the relative candidate link key.

Definition 13 (**Formal context expressing candidate link keys of two aligned classes**). Given two aligned classes c and c', we denote respectively by $D = \mathscr{T} \times \mathscr{P} \times \mathscr{V}$ and $D' = \langle \mathscr{T}', \mathscr{P}', \mathscr{V} \rangle$ the triple (instances, properties, values) of each class. From these notations, we build the formal context $\langle G, M, I \rangle$ with:

$G = \mathscr{T} \times \mathscr{T}'$ the set of pairs of instances,

$M = \{\exists, \forall, \forall\exists\} \times \mathscr{P} \times \mathscr{P}'$ the set of quantified pairs of properties.

I is such that,

$$\begin{aligned}
\langle o,o'\rangle I\langle \exists,p,p'\rangle &\text{ iff } \exists v; \langle o,p,v\rangle \in D \text{ and } \langle o',p',v\rangle \in D',\\
\langle o,o'\rangle I\langle \forall,p,p'\rangle &\text{ iff } \forall v; \langle o,p,v\rangle \in D \implies \langle o',p',v\rangle \in D'\\
&\quad \text{and } \forall v; \langle o',p',v\rangle \in D' \implies \langle o,p,v\rangle \in D\\
\langle o,o'\rangle I\langle \forall\exists,p,p'\rangle &\text{ iff } \exists v; \langle o,p,v\rangle \in D\\
&\quad \text{and } \forall v; \langle o,p,v\rangle \in D \implies \langle o',p',v\rangle \in D'\\
&\quad \text{and } \forall v; \langle o',p',v\rangle \in D' \implies \langle o,p,v\rangle \in D
\end{aligned}$$

The $\exists$ symbol expresses the IN-condition and similarly the $\forall$ and $\forall\exists$ symbols express EQ-condition and the EQ-IN-condition.[1] For two classes, thanks to this encoding, we are able to build the binary table representation of the link key problem and to use FCA algorithm to obtain a lattice and by the way the formal concepts which are candidate link keys.

In order to expose this encoding, we reuse the two ontologies `myLab` and `myCity` presented in Section 3.1. The first ontology contains the `myLab:researcher` class with the properties $p_1 =$ `myLab:HasEmail` and $p_2 =$ `myLab:isAuthor`. Its two instances are $t1 =$ `myLab:TomasDupond` and $t2 =$ `myLab:JulesMartin`. This class is aligned to the `myCity:Inhabitant` class of the second ontology which has two properties: a property $p'_1 =$ `myCity:HasEmail` and a property $p'_2 =$ `myCity:isAuthor`. Its instances are $t'_1 =$ `myCity:TomDupond` and $t'_2 =$ `myCity:JulesMartin`. We omit the $\forall\exists$ quantification which in this example obtains the same results than the $\forall$ quantification.

	(t_1,t'_1)	(t_2,t'_1)	(t_1,t'_2)	(t_2,t'_2)
$(\exists,p_1,p'_1)$	1	0	0	1
$(\forall,p_1,p'_1)$	0	0	0	1
$(\exists,p_1,p'_2)$	0	0	0	1
$(\forall,p_1,p'_2)$	0	0	0	0
$(\exists,p_2,p'_1)$	0	0	0	1
$(\forall,p_2,p'_1)$	0	0	0	0
$(\exists,p_2,p'_2)$	1	0	0	1
$(\forall,p_2,p'_2)$	1	0	0	1

Table 4.1 – FCA encoding for the classes `myLab:researcher` and `myCity:isAuthor`

Table 4.1 encodes the problem for the two given aligned classes. From this formal context, we can extract the formal concepts listed below. In spite of the smallness of the example, the extracted concept $C2$ has exactly the expected pairs of keys providing a perfect interlinking. The candidate link key is its intent: $(\exists,p_1,p'_1),(\exists,p_2,p'_2),(\forall,p_2,p'_2)$. The linked pairs of instances are its extent: $\{(t_1,t'_1),(t_2,t'_2)\}$

$$\begin{gathered}
C1 = (G,\{\}),\\
C2 = (\{(t_1,t'_1),(t_2,t'_2)\},\{(\exists,p_1,p'_1),(\exists,p_2,p'_2),(\forall,p_2,p'_2)\}),\\
C3 = (\{(t_2,t'_2)\},\{(\exists,p_1,p'_1),(\forall,p_1,p'_1),(\exists,p_1,p'_2),(\exists,p_2,p'_1),(\exists,p_2,p'_2),(\forall,p_2,p'_2)\}),\\
C4 = (\{\},M).
\end{gathered}$$

[1] The $\exists$, $\forall$ and $\forall\exists$ symbols of the encoding do not correspond to those present in the RCA scaling operators. The definitions of the previously presented scaling operators come from descriptors of description logic.

4.2 Ontology linking process

For the moment we are only able to perform data interlinking between instances that belongs to pairs of aligned classes. However, classes may rely on other classes and thus their link keys depend on other link keys. We propose to take a look at Figure 4.1. The represented graph contains a simple case study of two partially aligned ontologies. Each class of the first ontology is mapped to the second one.

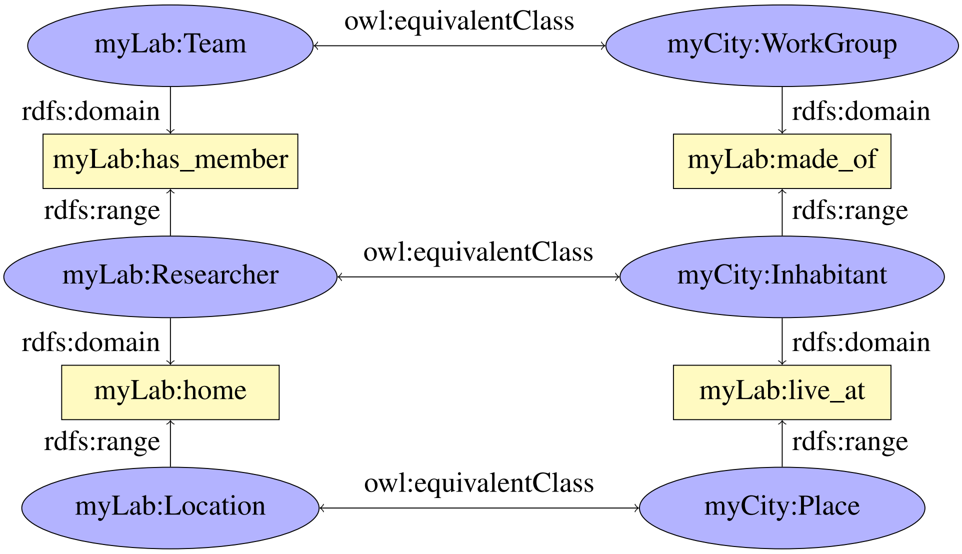


Figure 4.1 – RDFS graph of two aligned ontologies

Our proposal is to achieve data interlinking in a cascading way. We cannot directly build a formal context for the pair of classes `myLab:Researcher` and `myCity:Inhabitant`. These two classes depend respectively of the classes `myLab:Location` and `myCity:Place`. Hence, building a formal context for `myLab:Researcher` and `myCity:Inhabitant` requires to evaluate the quantified pairs involving the property `myLab:Home` and the property `myCity:live_at`. We propose to start by extracting candidate link keys for the classes having no dependency. From these extraction results, we are now able to deal with new classes, those only depending on the aligned pairs handled in the previous steps. Our procedure adopts a greedy approach. Iteratively, at each step, we extract candidate link keys for a specific alignment and select the best one. At the end of the process the two ontologies are completely interlinked. In this example, the process starts by determining the correspondences between the classes `myLab:Location` and `myCity:Place`, then between `myLab:Researcher` and `myCity:Inhabitant` and to finish between `myLab:Team` and `myCity:WorkGroup`.

A limitation of this method is that the dependencies induced by the pairs of classes should not form cycle. Such a situation cannot be handled because the process cannot determine a starting point. A possible solution would be to ignore the dependency of a first pair of classes chosen randomly. However such a choice implies the loss of part of the information. We discuss in more detail about the dependency problem in the next chapter.

4.3 FCA process example

We present below an example of unsupervised selection of link key. We reuse the previous example which is composed of two ontologies: the ontology `myLab` on the left and the ontology `myCity` on the right sharing values on the middle (see Figure 4.2). As mentioned, we know that the classes `myLab:Researcher` and `myCity:Inhabitant` are aligned, as well as `myLab:Location` and `myCity:Place`. None of the instances of this example hold multiple values for a given property. We made this choice to simplify the explanation. As the ∀∃ quantification is a specialization of both the ∀ and the ∃ quantifications, we omit these two when the ∀∃ quantification holds.

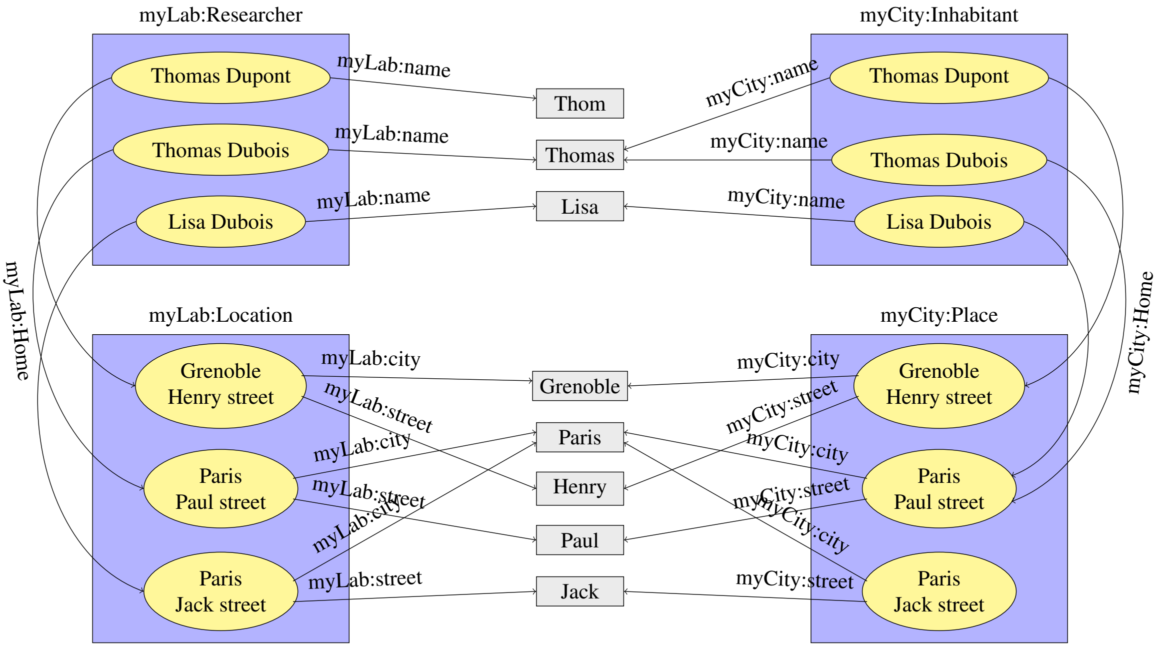


Figure 4.2 – RDF Graph of two ontologies

The first step is to take a look at the dependencies. As we can see, the pair of aligned classes with no dependency is `myLab:Location` and `myCity:Place`. Hence, we are able to build the formal context of these two classes. We extract the formal concepts from it and to finish, we evaluate the candidate link key of each extent to select the best link key according to the f-measures or the h-mean depending on whether we perform supervised or unsupervised evaluations. In this example, the extracted link key is {(∀∃,`myLab:street`,`myCity:street`), (∀∃,`myLab:city`,`myCity:city`)}. The interlinking is perfect with a coverage and a discriminability of 1. From this result, we can repeat the process in order to interlink the two classes `myLab:Researcher` and `myCity:Inhabitant`. They have two properties, their name and their home. Thanks to the first step, we are able to compare the home property despite of the fact that it is not a primitive type. The process gives to us two equivalent link keys with a h-mean of 0.74: the link key (∀∃,`myLab:home`,`myCity:home`) and the link key (∀∃ ,`myLab:name`,`myCity:name`) .

Global selection of link keys

The previous example raises a problem. Indeed, the proposed process aims at selecting the best link keys at the current iteration in order to use them at the next step. This greedy approach is not always optimal and particularly in this small example. The term ‘best’ refers to the candidate link key getting the higher h-mean (equivalently the higher f-measure) for the given pair of classes. However, at the next iteration, this link key is not necessary the best anymore for the new pairs of classes. This is what we can exhibit from the example. Using the link keys $\{(\forall\exists,\texttt{myLab:street},\texttt{myCity:street}),(\forall\exists,\texttt{myLab:city},\texttt{myLab:city})\}$ and the link key $\{(\forall\exists,\texttt{myLab:home},\texttt{myCity:home})\}$ give to us a h-mean of 0.74 on the second pair of classes. On the other hand, using the link key $\{(\forall\exists,\texttt{myLab:city},\texttt{myCity:city})\}$ which is not the best at the first iteration lets us get a score of 0.80 at the second iteration with the link key $\{(\forall\exists,\texttt{myLab:home},\texttt{myCity:home})\}$.

In order to fix it, a possibility is to modify our proposed algorithm. Instead of selecting only one link key and not considering the others, one solution is to increase the search space. At each step, a solution is to extract all the link keys and produce the next iterations for all of them. Hence, at the end, by performing a global evaluation, we are able to select the set of link keys maximising our objective. This approach has some drawbacks: a significant increase of the computational cost and of the memory use.

4.4 Conclusion

In this chapter, we saw the FCA encoding of the link key extraction problem for ontologies where aligned classes are known. We particularly saw how to deal with the EQ and IN conditions in the encoding of pair of aligned classes. We also established a way of iterating the process in order to link the two ontologies via a greedy approach. Object properties are supported as well as datatype properties. To conclude, we remarked that this approach have some drawbacks such as it does not support circular dependency and that the extracted link keys are not always optimal.

The next chapter aims at dealing with these limitations of the presented FCA encoding. We firstly discuss about circular dependencies and how they hold. Then, we propose a new encoding using RCA that is able to go beyond the limits of FCA by dealing with circular dependencies. Moreover this encoding does not require any alignment information and any greedy approach.

— 5 —
Data interlinking with RCA

The previous section has presented a way of interlinking ontologies under two assumptions. The first assumption is that classes of the two ontologies are aligned. One way of dealing with the lack of alignment is to perform a preprocessing step using state of the art techniques. However, these techniques are costly and not error free. The second assumption is that dependencies of the classes do not hold cycle. This is a strong issue considering daily life data set. The first section of this chapter analyses the issue. The next section aims at proposing a way of dealing with it by introducing a new encoding. For that purpose, we use relational context analysis to express the properties holding dependencies via relational contexts. Indeed, the RCA algorithm converges even if relations are circular. Hence, as we will see, our proposed RCA encoding does not require any alignment and any dependency consideration anymore.

5.1 Dependency problem

Two classes are often dependent on each other. A property expressing a statement is often accompanied by a second link expressing the reciprocal one. Figure 5.1 is an example of graph containing circular dependencies. It reuses the two ontologies `myLab` and `myCity` studied in the previous chapter. As example we look at the properties `myLab:work_in` and `myLab:has_member`. The properties are dual considering the dependencies held as well as the expressed meaning. A possible workaround is to perform the process with only one of the two links and to use the extracted link key for both links. However, choosing to classify employees in function of their job or jobs in function of the employee would orient the result. Moreover, more complex behaviours can be identified where such bypass cannot be used.

The first problematic situation is about two links holding a dependency cycle that do not express a reciprocal meaning. This is the case of the properties `myCity:live_at` and `myCity:is_owned_by`. Removing one of the two links is like removing some parts of the information. Hence, the workaround does not work in this situation.

Another problematic case is due to classes depending on themselves. `myLab:Researcher` is one of them. It depends to itself because of the property `myLab:is_friend_of`. This kind of relations cannot be ignored. As one can imagine, in this example, one way of identifying people is to identify their relationship.

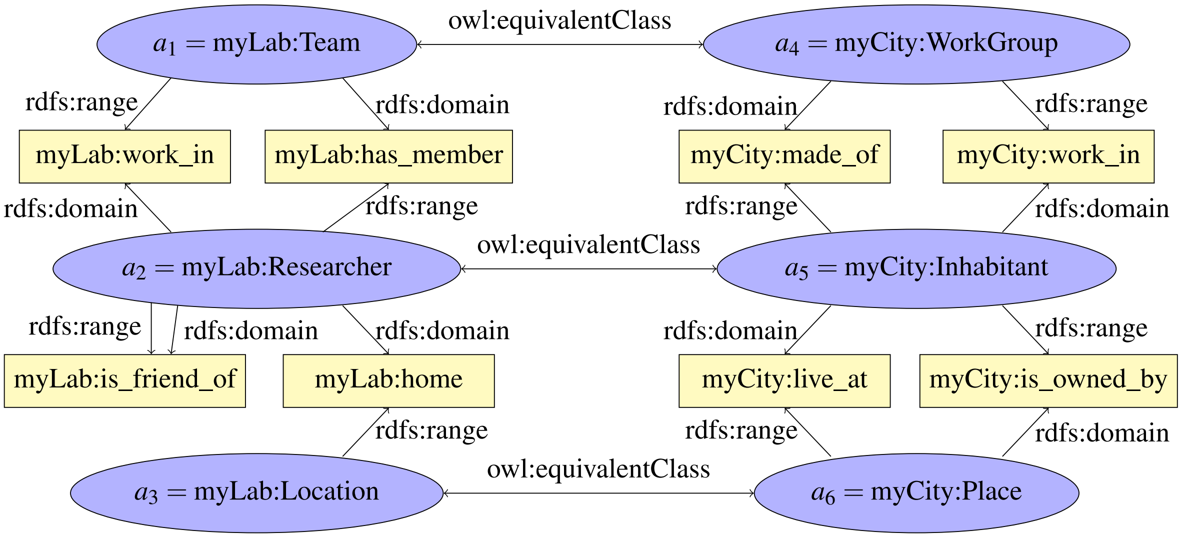


Figure 5.1 – RDFS graph of aligned ontologies with circular dependencies

5.2 RCA encoding

The RCA algorithm assures convergence even if the relational context creates circular dependency. Hence, we propose a second encoding where relations are expressed thanks to relational contexts. In this way, we are able to ignore the alignments of the classes and to not take care about possible dependency cycles. For that purpose, the method is to build a context for all the possible pairs of classes. Even if we do not know the equivalent classes, we can deduce from the final results pairs of classes that only obtain bad candidate link keys (i.e. link keys with low f-measure or low h-mean) correspond to wrong alignments.

In our FCA encoding, objects of formal contexts are pairs of instances characterised by quantified pairs of properties. In order to elaborate our RCA encoding, we reuse the idea of simply restricting the properties. Our proposal is to build formal contexts with pairs of instances as objects and quantified pairs of datatype properties as attributes. On the other hand, object properties are expressed thanks to relational contexts. Considering this model, the formal contexts aim to extract intra-concepts. We are able to extract candidate link keys composed of datatype properties from them. At the same time, each pair of object properties has its own relational contexts to propagate the extracted intra-concepts, i.e., the candidate link keys initially composed of quantified pairs of datatype properties. After some iterations of the RCA algorithm, via the propagation process, datatype properties and object properties are mixed to compose as expected candidate link keys made of all the properties. We define below in more detail the two encodings.

Formal contexts encoding

Definition 14 (Formal context expressing candidate link keys for two classes). For all pairs of classes $(c,c') \in \mathscr{O} \times \mathscr{O}'$, we respectively denote by $D(c) \subseteq \mathscr{T}(c) \times \mathscr{P}(c) \times \mathscr{V}(c)$ the triple (instances, properties, values) for the class c. We also denote by $\mathscr{P}_v(c)$ the subset of datatype properties of $\mathscr{P}(c)$. The formal context $K_{c,c'} = \langle G,M,I \rangle$ is defined by:

$G = \mathscr{T}(c) \times \mathscr{T}'(c')$ the set of pairs of instances,

$M = \{\exists, \forall, \forall\exists\} \times \mathscr{P}_v(c) \times \mathscr{P}'_v(c)$ the set of quantified pairs of datatype properties.
I is such that,

$$\begin{aligned}
\langle o,o'\rangle I \langle \exists, p, p'\rangle &\text{ iff } \exists v; \langle o,p,v\rangle \in D \text{ and } \langle o',p',v\rangle \in D',\\
\langle o,o'\rangle I \langle \forall, p, p'\rangle &\text{ iff } \forall v; \langle o,p,v\rangle \in D \implies \langle o',p',v\rangle \in D'\\
&\qquad \text{and } \forall v; \langle o',p',v\rangle \in D' \implies \langle o,p,v\rangle \in D\\
\langle o,o'\rangle I \langle \forall\exists, p, p'\rangle &\text{ iff } \exists v; \langle o,p,v\rangle \in D\\
&\qquad \text{and } \forall v; \langle o,p,v\rangle \in D \implies \langle o',p',v\rangle \in D'\\
&\qquad \text{and } \forall v; \langle o',p',v\rangle \in D' \implies \langle o,p,v\rangle \in D
\end{aligned}$$

This encoding is similar to the one defined during the FCA encoding section. It is simply restricted to properties linked to primitive values and not to instances. One can notice that the example of the Figure 5.1 requires building nine contexts: a context for each couple of $\{a_1, a_2, a_3\} \times \{a_4, a_5, a_6\}$.

Relational contexts encoding

We choose to build one relational context for each pair of object properties given a pair of classes and to use one scaling operator for each quantification. We respectively choose the existential operator, the universal operator and the universal strict operator to model the IN, EQ and EQ-IN conditions. We denote by (p_1, p_2) a pair of object properties and by (c, c') the two relative classes being subjects of the triples and by (d, d') the two relative classes being objects.

Relational contexts are composed by three elements: source-objects, target-objects and by an incidence relation between them. We choose as target-objects the objects of the formal context representing (c, c'), i.e., pairs of instances of (c, c'). Similarly, we choose as source-objects the objects of the formal context representing (d, d'), i.e., pairs of instances of (d, d'). The incidence relation between the two sets is simply filled thanks to the data, 1 if a relation is held by (p_1, p_2) given the pair of instances of (c, c') and the pairs of instances of (d, d') otherwise 0.

Definition 15 (Relational context for a pair of object properties). The relational context $R_{p_1,p_2} = \langle G, M, I\rangle$ which puts in relation the objects of $K_{c,c'}$ and the objects of $K_{d,d'}$ is defined by:

$G_t = \mathscr{T}(c) \times \mathscr{T}(c')$ the set of pairs of instances of c and c'
$G_s = \mathscr{T}(d) \times \mathscr{T}'(d')$ the set of pairs of instances of d and d',
I is such that

$$\langle o_1, o_2\rangle I \langle o_3, o_4\rangle \text{ iff } \langle o_1, p_1, o_3\rangle \in D(c) \text{ and } \langle o_2, p_2, o_4\rangle \in D(c')$$

The scaling operator propagates concepts and introduces the quantified pairs of object properties. Iteratively, the quantified pairs of object properties accompanied by the relative concepts are added to the attributes of the initial formal contexts,, i.e.,, the quantified pairs of datatype properties. Hence, the final candidate link keys are as expected composed by pairs of both datatype properties and propagated quantified pairs of object properties.

5.3 RCA process example

In order to illustrate our new encoding, the last example of Figure 5.1 is reused by making it more complex. Figure 5.2 illustrates this new case study. We firstly remove all the information about the alignments, i.e., `owl:equivalentClass` links. Hence, we do not have any information about the alignment of the classes. We secondly make more complex the ontology by adding the last name information and by creating dependency cycles . Our first addition is to add a circular dependency to the class `myLab:Researcher` via the addition of the property `myLab:sibling`. Moreover, we also add two properties: the property `myLab:owned_by` between the classes `myLab:Researcher` and `myLab:Location` and the property `mycity:owned_by` between `mycity:Inhabitant` and `myCity:Place`. In the above graphs, we respectively denote by `fn`, `ln`, `ct` and `st`, the datatype properties `firstName`, `lastName`, `city` and `street`. We also respectively denote by `hm`, `sb` and `ow` the object properties `home`, `sibling` and `owned_by`.

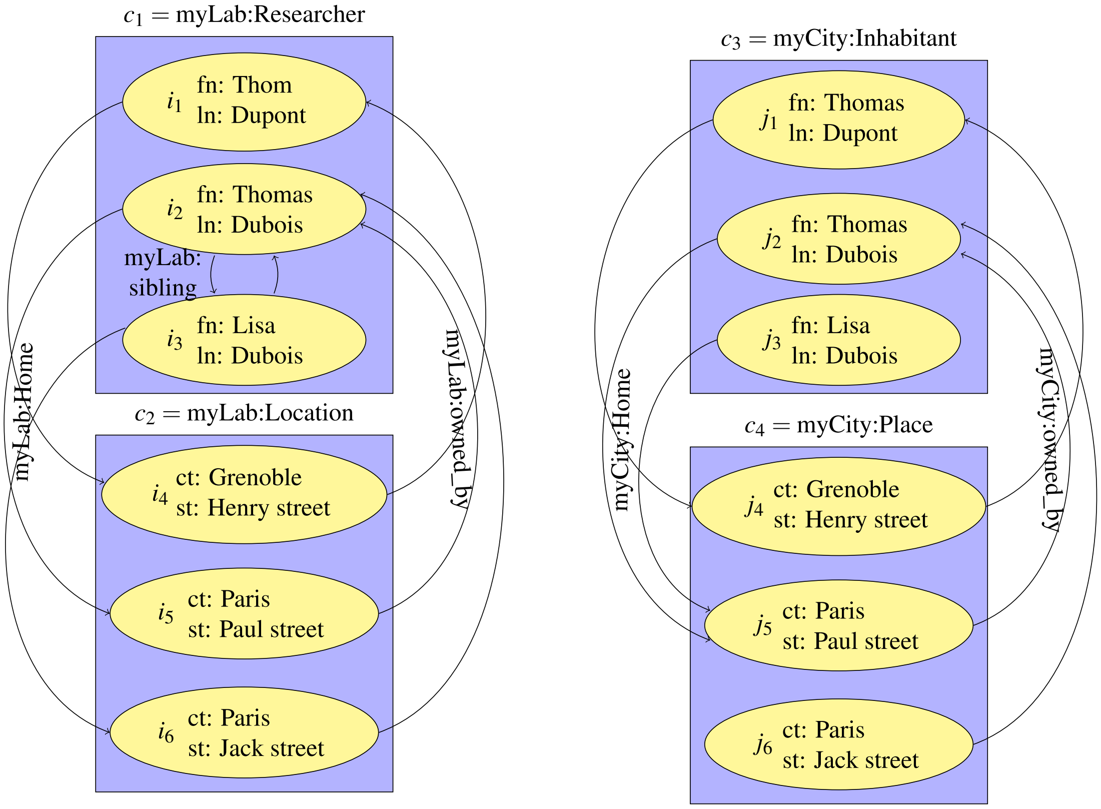

$c_1 =$ myLab:Researcher
i_1 fn: Thom ln: Dupont
i_2 fn: Thomas ln: Dubois
myLab: sibling
i_3 fn: Lisa ln: Dubois
$c_2 =$ myLab:Location
i_4 ct: Grenoble st: Henry street
i_5 ct: Paris st: Paul street
i_6 ct: Paris st: Jack street
myLab:Home
myLab:owned_by
$c_3 =$ myCity:Inhabitant
j_1 fn: Thomas ln: Dupont
j_2 fn: Thomas ln: Dubois
j_3 fn: Lisa ln: Dubois
$c_4 =$ myCity:Place
j_4 ct: Grenoble st: Henry street
j_5 ct: Paris st: Paul street
j_6 ct: Paris st: Jack street
myCity:Home
myCity:owned_by

Figure 5.2 – Case study 1: RDF graph of `myLab` on the left and `myCity` on the right

Our method is to build an initial formal context for each pair of classes that only uses datatype properties. Hence, we have to build the contexts K_{c_1,c_3}, K_{c_1,c_4}, K_{c_2,c_3} and K_{c_2,c_4}. Additionally, we have to build one relational context for each pair of object properties: $R_{\texttt{hm},\texttt{hm}}$, $R_{\texttt{sb},\texttt{hm}}$, $R_{\texttt{ow},\texttt{hm}}$, $R_{\texttt{hm},\texttt{ow}}$, $R_{\texttt{sb},\texttt{ow}}$ and $R_{\texttt{ow},\texttt{ow}}$. The cross tables of these contexts are available below. We omit the contexts K_{c_1,c_4}, K_{c_2,c_3}, , $R_{\texttt{sb},\texttt{hm}}$, $R_{\texttt{ow},\texttt{hm}}$, $R_{\texttt{hm},\texttt{ow}}$ and $R_{\texttt{sb},\texttt{ow}}$ because their tables are completely filled by 0. They correspond to obvious bad classes and properties alignments.

K_{c_1,c_3}	(i_2,j_2)	(i_2,j_3)	(i_2,j_1)	(i_1,j_2)	(i_1,j_3)	(i_1,j_1)	(i_3,j_2)	(i_3,j_3)	(i_3,j_1)
$(\exists,\texttt{ln},\texttt{fn})$	0	0	0	0	0	0	0	0	0
$(\exists,\texttt{ln},\texttt{ln})$	1	1	0	0	0	1	1	1	0
$(\exists,\texttt{fn},\texttt{fn})$	1	0	1	0	0	0	0	1	0
$(\exists,\texttt{fn},\texttt{ln})$	0	0	0	0	0	0	0	0	0
$(\forall,\texttt{ln},\texttt{fn})$	0	0	0	0	0	0	0	0	0
$(\forall,\texttt{ln},\texttt{ln})$	1	1	0	0	0	1	1	1	0
$(\forall,\texttt{fn},\texttt{fn})$	1	0	1	0	0	0	0	1	0
$(\forall,\texttt{fn},\texttt{ln})$	0	0	0	0	0	0	0	0	0
$(\forall\exists,\texttt{ln},\texttt{fn}$	0	0	0	0	0	0	0	0	0
$(\forall\exists,\texttt{ln},\texttt{ln}$	1	1	0	0	0	1	1	1	0
$(\forall\exists,\texttt{fn},\texttt{fn}$	1	0	1	0	0	0	0	1	0
$(\forall\exists,\texttt{fn},\texttt{ln}$	0	0	0	0	0	0	0	0	0

K_{c_2,c_4}	(i_6,j_5)	(i_6,j_4)	(i_6,j_6)	(i_4,j_5)	(i_4,j_4)	(i_4,j_6)	(i_5,j_5)	(i_5,j_4)	(i_5,j_6)
$(\exists,\texttt{st},\texttt{st})$	0	0	1	0	1	0	1	0	0
$(\exists,\texttt{st},\texttt{ct})$	0	0	0	0	0	0	0	0	0
$(\exists,\texttt{ct},\texttt{st})$	0	0	0	0	0	0	0	0	0
$(\exists,\texttt{ct},\texttt{ct})$	1	0	1	0	1	0	1	0	1
$(\forall,\texttt{st},\texttt{st})$	0	0	1	0	1	0	1	0	0
$(\forall,\texttt{st},\texttt{ct})$	0	0	0	0	0	0	0	0	0
$(\forall,\texttt{ct},\texttt{st})$	0	0	0	0	0	0	0	0	0
$(\forall,\texttt{ct},\texttt{ct})$	1	0	1	0	1	0	1	0	1
$(\forall\exists,\texttt{st},\texttt{st})$	0	0	1	0	1	0	1	0	0
$(\forall\exists,\texttt{st},\texttt{ct})$	0	0	0	0	0	0	0	0	0
$(\forall\exists,\texttt{ct},\texttt{st})$	0	0	0	0	0	0	0	0	0
$(\forall\exists,\texttt{ct},\texttt{ct})$	1	0	1	0	1	0	1	0	1

Figure 5.3 – Formal contexts K_{c_1,c_3} and K_{c_2,c_4}

$(\texttt{hm},\texttt{hm})$	(i_2,j_2)	(i_2,j_3)	(i_2,j_1)	(i_1,j_2)	(i_1,j_3)	(i_1,j_1)	(i_3,j_2)	(i_3,j_3)	(i_3,j_1)
(i_6,j_5)	0	0	0	0	0	0	1	1	0
(i_6,j_4)	0	0	0	0	0	0	0	0	1
(i_6,j_6)	0	0	0	0	0	0	0	0	0
(i_4,j_5)	0	0	0	1	1	0	0	0	0
(i_4,j_4)	0	0	0	0	0	1	0	0	0
(i_4,j_6)	0	0	0	0	0	0	0	0	0
(i_5,j_5)	1	1	0	0	0	0	0	0	0
(i_5,j_4)	0	0	1	0	0	0	0	0	0
(i_5,j_6)	0	0	0	0	0	0	0	0	0

$(\texttt{ow},\texttt{ow})$	(i_6,j_5)	(i_6,j_4)	(i_6,j_6)	(i_4,j_5)	(i_4,j_4)	(i_4,j_6)	(i_5,j_5)	(i_5,j_4)	(i_5,j_6)
(i_2,j_2)	1	0	1	0	0	0	1	0	1
(i_2,j_3)	0	0	0	0	0	0	0	0	0
(i_2,j_1)	0	1	0	0	0	0	0	1	0
(i_1,j_2)	0	0	0	1	0	1	0	0	0
(i_1,j_3)	0	0	0	0	0	0	0	0	0
(i_1,j_1)	0	0	0	0	1	0	0	0	0
(i_3,j_2)	0	0	0	0	0	0	0	0	0
(i_3,j_3)	0	0	0	0	0	0	0	0	0
(i_3,j_1)	0	0	0	0	0	0	0	0	0

Figure 5.4 – Relational contexts $R_{\texttt{hm},\texttt{hm}}$ and $R_{\texttt{ow},\texttt{ow}}$

From these contexts, we apply the RCA algorithm. The first step builds the lattices of the contexts K_{c_1,c_3} and K_{c_2,c_4} which are displayed below. They are composed of concepts having as intent candidate link key made of datatype properties. The extents are the linked instances.

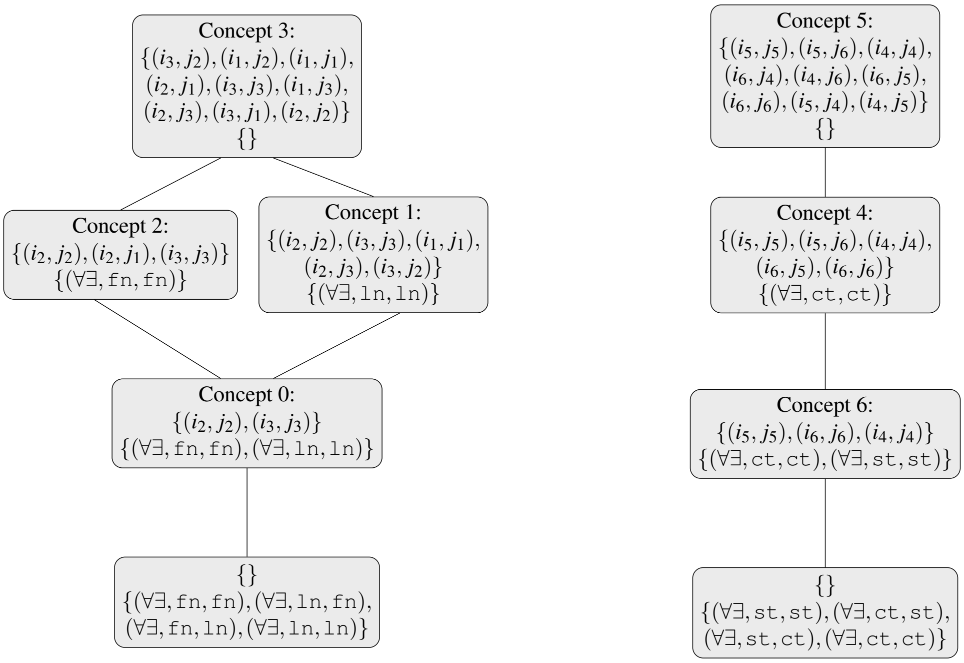

Concept 3:
$\{(i_3,j_2),(i_1,j_2),(i_1,j_1),(i_2,j_1),(i_3,j_3),(i_1,j_3),(i_2,j_3),(i_3,j_1),(i_2,j_2)\}$
$\{\}$
Concept 2:
$\{(i_2,j_2),(i_2,j_1),(i_3,j_3)\}$
$\{(\forall\exists,\mathtt{fn},\mathtt{fn})\}$
Concept 1:
$\{(i_2,j_2),(i_3,j_3),(i_1,j_1),(i_2,j_3),(i_3,j_2)\}$
$\{(\forall\exists,\mathtt{ln},\mathtt{ln})\}$
Concept 0:
$\{(i_2,j_2),(i_3,j_3)\}$
$\{(\forall\exists,\mathtt{fn},\mathtt{fn}),(\forall\exists,\mathtt{ln},\mathtt{ln})\}$
$\{\}$
$\{(\forall\exists,\mathtt{fn},\mathtt{fn}),(\forall\exists,\mathtt{ln},\mathtt{fn}),(\forall\exists,\mathtt{fn},\mathtt{ln}),(\forall\exists,\mathtt{ln},\mathtt{ln})\}$
Concept 5:
$\{(i_5,j_5),(i_5,j_6),(i_4,j_4),(i_6,j_4),(i_4,j_6),(i_6,j_5),(i_6,j_6),(i_5,j_4),(i_4,j_5)\}$
$\{\}$
Concept 4:
$\{(i_5,j_5),(i_5,j_6),(i_4,j_4),(i_6,j_5),(i_6,j_6)\}$
$\{(\forall\exists,\mathtt{ct},\mathtt{ct})\}$
Concept 6:
$\{(i_5,j_5),(i_6,j_6),(i_4,j_4)\}$
$\{(\forall\exists,\mathtt{ct},\mathtt{ct}),(\forall\exists,\mathtt{st},\mathtt{st})\}$
$\{\}$
$\{(\forall\exists,\mathtt{st},\mathtt{st}),(\forall\exists,\mathtt{ct},\mathtt{st}),(\forall\exists,\mathtt{st},\mathtt{ct}),(\forall\exists,\mathtt{ct},\mathtt{ct})\}$

Figure 5.5 – Iteration 0: On the left the lattice of K_{c_1,c_3} and on the right the lattice of K_{c_2,c_4}

The second step of the algorithm builds the scaled contexts. For that purpose we use the built lattices to create new attributes from the contexts having relations. These new attributes correspond to candidate link keys made of pairs of object properties given some relative candidate link keys of other pair of classes. We have to propagate the concepts by concatenating the scaled contexts. Table 5.1 shows the scaled contexts resulting from the first iteration.

R'_{c_1,c_3}	(i_2,j_2)	(i_2,j_3)	(i_2,j_1)	(i_1,j_2)	(i_1,j_3)	(i_1,j_1)	(i_3,j_2)	(i_3,j_3)	(i_3,j_1)
$(\forall\exists,\mathtt{hm},\mathtt{hm}):C4$	1	1	0	0	0	1	1	1	0
$(\forall\exists,\mathtt{hm},\mathtt{hm}):C5$	1	1	1	1	1	1	1	1	1
$(\forall\exists,\mathtt{hm},\mathtt{hm}):C6$	1	1	0	0	0	1	0	0	0

R'_{c_2,c_4}	(i_6,j_5)	(i_6,j_4)	(i_6,j_6)	(i_4,j_5)	(i_4,j_4)	(i_4,j_6)	(i_5,j_5)	(i_5,j_4)	(i_5,j_6)
$(\forall\exists,\mathtt{ow},\mathtt{ow}):C0$	1	0	1	0	0	0	1	0	1
$(\forall\exists,\mathtt{ow},\mathtt{ow}):C1$	1	0	1	0	1	0	1	0	1
$(\forall\exists,\mathtt{ow},\mathtt{ow}):C2$	1	1	1	0	0	0	1	1	1
$(\forall\exists,\mathtt{ow},\mathtt{ow}):C3$	1	1	1	1	1	1	1	1	1

Table 5.1 – Iteration 1: R'_{c_1,c_2} and R'_{c_2,c_4}

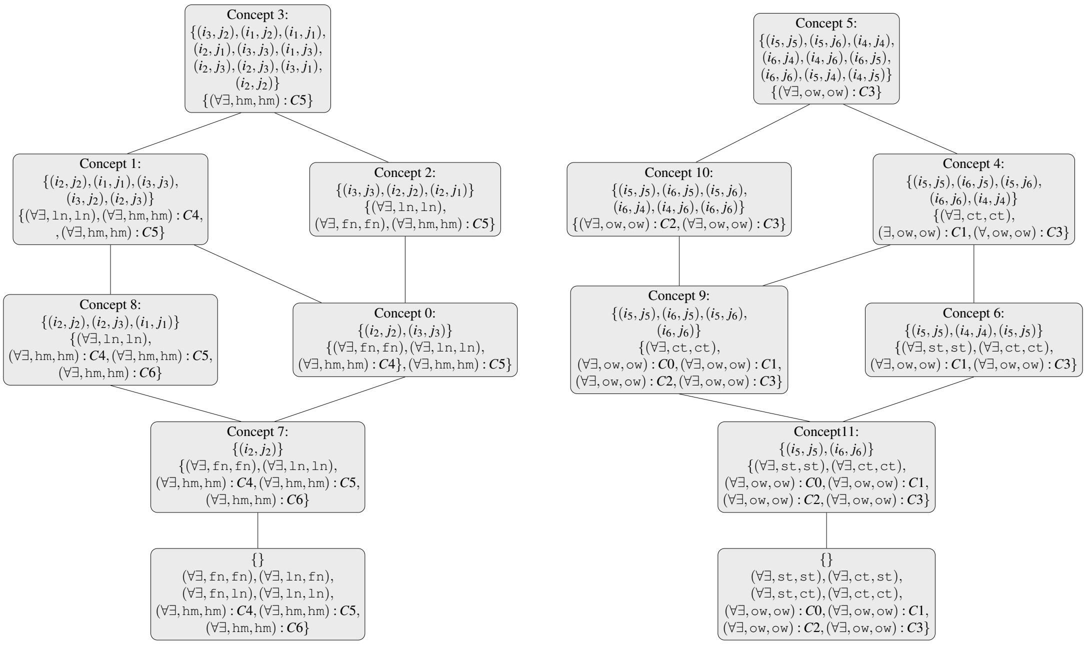

Concept 3:
$\{(i_3,j_2),(i_1,j_2),(i_1,j_1),$
$(i_2,j_1),(i_3,j_3),(i_1,j_3),$
$(i_2,j_3),(i_2,j_3),(i_3,j_1),$
$(i_2,j_2)\}$
$\{(\forall\exists,\mathrm{hm},\mathrm{hm}):C5\}$
Concept 1:
$\{(i_2,j_2),(i_1,j_1),(i_3,j_3),$
$(i_3,j_2),(i_2,j_3)\}$
$\{(\forall\exists,\texttt{ln},\texttt{ln}),(\forall\exists,\mathrm{hm},\mathrm{hm}):C4,$
$,(\forall\exists,\mathrm{hm},\mathrm{hm}):C5\}$
Concept 2:
$\{(i_3,j_3),(i_2,j_2),(i_2,j_1)\}$
$\{(\forall\exists,\texttt{ln},\texttt{ln}),$
$(\forall\exists,\texttt{fn},\texttt{fn}),(\forall\exists,\mathrm{hm},\mathrm{hm}):C5\}$
Concept 8:
$\{(i_2,j_2),(i_2,j_3),(i_1,j_1)\}$
$\{(\forall\exists,\texttt{ln},\texttt{ln}),$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C4,(\forall\exists,\mathrm{hm},\mathrm{hm}):C5,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C6\}$
Concept 0:
$\{(i_2,j_2),(i_3,j_3)\}$
$\{(\forall\exists,\texttt{fn},\texttt{fn}),(\forall\exists,\texttt{ln},\texttt{ln}),$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C4\},(\forall\exists,\mathrm{hm},\mathrm{hm}):C5\}$
Concept 7:
$\{(i_2,j_2)\}$
$\{(\forall\exists,\texttt{fn},\texttt{fn}),(\forall\exists,\texttt{ln},\texttt{ln}),$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C4,(\forall\exists,\mathrm{hm},\mathrm{hm}):C5,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C6\}$
$\{\}$
$(\forall\exists,\texttt{fn},\texttt{fn}),(\forall\exists,\texttt{ln},\texttt{fn}),$
$(\forall\exists,\texttt{fn},\texttt{ln}),(\forall\exists,\texttt{ln},\texttt{ln}),$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C4,(\forall\exists,\mathrm{hm},\mathrm{hm}):C5,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C6\}$
Concept 5:
$\{(i_5,j_5),(i_5,j_6),(i_4,j_4),$
$(i_6,j_4),(i_4,j_6),(i_6,j_5),$
$(i_6,j_6),(i_5,j_4),(i_4,j_5)\}$
$\{(\forall\exists,\texttt{ow},\texttt{ow}):C3\}$
Concept 10:
$\{(i_5,j_5),(i_6,j_5),(i_5,j_6),$
$(i_6,j_4),(i_4,j_6),(i_6,j_6)\}$
$\{(\forall\exists,\texttt{ow},\texttt{ow}):C2,(\forall\exists,\texttt{ow},\texttt{ow}):C3\}$
Concept 4:
$\{(i_5,j_5),(i_6,j_5),(i_5,j_6),$
$(i_6,j_6),(i_4,j_4)\}$
$\{(\forall\exists,\texttt{ct},\texttt{ct}),$
$(\exists,\texttt{ow},\texttt{ow}):C1,(\forall,\texttt{ow},\texttt{ow}):C3\}$
Concept 9:
$\{(i_5,j_5),(i_6,j_5),(i_5,j_6),$
$(i_6,j_6)\}$
$\{(\forall\exists,\texttt{ct},\texttt{ct}),$
$(\forall\exists,\texttt{ow},\texttt{ow}):C0,(\forall\exists,\texttt{ow},\texttt{ow}):C1,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C2,(\forall\exists,\texttt{ow},\texttt{ow}):C3\}$
Concept 6:
$\{(i_5,j_5),(i_4,j_4),(i_5,j_5)\}$
$\{(\forall\exists,\texttt{st},\texttt{st}),(\forall\exists,\texttt{ct},\texttt{ct}),$
$(\forall\exists,\texttt{ow},\texttt{ow}):C1,(\forall\exists,\texttt{ow},\texttt{ow}):C3\}$
Concept11:
$\{(i_5,j_5),(i_6,j_6)\}$
$\{(\forall\exists,\texttt{st},\texttt{st}),(\forall\exists,\texttt{ct},\texttt{ct}),$
$(\forall\exists,\texttt{ow},\texttt{ow}):C0,(\forall\exists,\texttt{ow},\texttt{ow}):C1,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C2,(\forall\exists,\texttt{ow},\texttt{ow}):C3\}$
$\{\}$
$(\forall\exists,\texttt{st},\texttt{st}),(\forall\exists,\texttt{ct},\texttt{st}),$
$(\forall\exists,\texttt{st},\texttt{ct}),(\forall\exists,\texttt{ct},\texttt{ct}),$
$(\forall\exists,\texttt{ow},\texttt{ow}):C0,(\forall\exists,\texttt{ow},\texttt{ow}):C1,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C2,(\forall\exists,\texttt{ow},\texttt{ow}):C3\}$

Figure 5.6 – Iteration 1: On the left the lattice of K_{c_1,c_3} and on the right the lattice of K_{c_2,c_4}

From these two tables, we repeat the process by building the corresponding lattices. This example converges after 4 iterations to the lattices of Figure 5.7. This is from these two lattices that we evaluate candidate link keys and select the final link keys. The details of the process are explained in the next section.

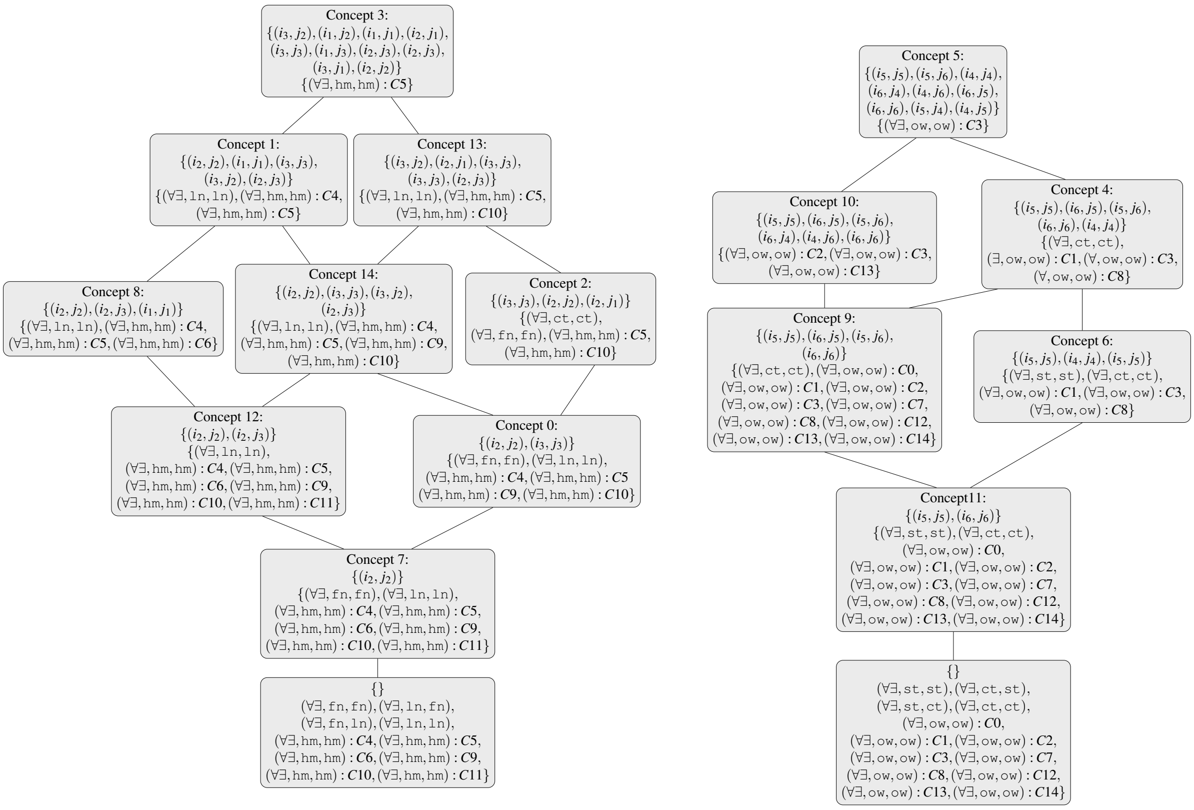

Concept 3:
$\{(i_3,j_2),(i_1,j_2),(i_1,j_1),(i_2,j_1),$
$(i_3,j_3),(i_1,j_3),(i_2,j_3),(i_2,j_3),$
$(i_3,j_1),(i_2,j_2)\}$
$\{(\forall\exists,\mathrm{hm},\mathrm{hm}):C5\}$
Concept 1:
$\{(i_2,j_2),(i_1,j_1),(i_3,j_3),$
$(i_3,j_2),(i_2,j_3)\}$
$\{(\forall\exists,\texttt{ln},\texttt{ln}),(\forall\exists,\mathrm{hm},\mathrm{hm}):C4,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C5\}$
Concept 13:
$\{(i_3,j_2),(i_2,j_1),(i_3,j_3),$
$(i_3,j_3),(i_2,j_3)\}$
$\{(\forall\exists,\texttt{ln},\texttt{ln}),(\forall\exists,\mathrm{hm},\mathrm{hm}):C5,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C10\}$
Concept 8:
$\{(i_2,j_2),(i_2,j_3),(i_1,j_1)\}$
$\{(\forall\exists,\texttt{ln},\texttt{ln}),(\forall\exists,\mathrm{hm},\mathrm{hm}):C4,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C5,(\forall\exists,\mathrm{hm},\mathrm{hm}):C6\}$
Concept 14:
$\{(i_2,j_2),(i_3,j_3),(i_3,j_2),$
$(i_2,j_3)\}$
$\{(\forall\exists,\texttt{ln},\texttt{ln}),(\forall\exists,\mathrm{hm},\mathrm{hm}):C4,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C5,(\forall\exists,\mathrm{hm},\mathrm{hm}):C9,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C10\}$
Concept 2:
$\{(i_3,j_3),(i_2,j_2),(i_2,j_1)\}$
$\{(\forall\exists,\texttt{ct},\texttt{ct}),$
$(\forall\exists,\texttt{fn},\texttt{fn}),(\forall\exists,\mathrm{hm},\mathrm{hm}):C5,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C10\}$
Concept 12:
$\{(i_2,j_2),(i_2,j_3)\}$
$\{(\forall\exists,\texttt{ln},\texttt{ln}),$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C4,(\forall\exists,\mathrm{hm},\mathrm{hm}):C5,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C6,(\forall\exists,\mathrm{hm},\mathrm{hm}):C9,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C10,(\forall\exists,\mathrm{hm},\mathrm{hm}):C11\}$
Concept 0:
$\{(i_2,j_2),(i_3,j_3)\}$
$\{(\forall\exists,\texttt{fn},\texttt{fn}),(\forall\exists,\texttt{ln},\texttt{ln}),$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C4,(\forall\exists,\mathrm{hm},\mathrm{hm}):C5$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C9,(\forall\exists,\mathrm{hm},\mathrm{hm}):C10\}$
Concept 7:
$\{(i_2,j_2)\}$
$\{(\forall\exists,\texttt{fn},\texttt{fn}),(\forall\exists,\texttt{ln},\texttt{ln}),$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C4,(\forall\exists,\mathrm{hm},\mathrm{hm}):C5,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C6,(\forall\exists,\mathrm{hm},\mathrm{hm}):C9,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C10,(\forall\exists,\mathrm{hm},\mathrm{hm}):C11\}$
$\{\}$
$(\forall\exists,\texttt{fn},\texttt{fn}),(\forall\exists,\texttt{ln},\texttt{fn}),$
$(\forall\exists,\texttt{fn},\texttt{ln}),(\forall\exists,\texttt{ln},\texttt{ln}),$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C4,(\forall\exists,\mathrm{hm},\mathrm{hm}):C5,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C6,(\forall\exists,\mathrm{hm},\mathrm{hm}):C9,$
$(\forall\exists,\mathrm{hm},\mathrm{hm}):C10,(\forall\exists,\mathrm{hm},\mathrm{hm}):C11\}$
Concept 5:
$\{(i_5,j_5),(i_5,j_6),(i_4,j_4),$
$(i_6,j_4),(i_4,j_6),(i_6,j_5),$
$(i_6,j_6),(i_5,j_4),(i_4,j_5)\}$
$\{(\forall\exists,\texttt{ow},\texttt{ow}):C3\}$
Concept 10:
$\{(i_5,j_5),(i_6,j_5),(i_5,j_6),$
$(i_6,j_4),(i_4,j_6),(i_6,j_6)\}$
$\{(\forall\exists,\texttt{ow},\texttt{ow}):C2,(\forall\exists,\texttt{ow},\texttt{ow}):C3,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C13\}$
Concept 4:
$\{(i_5,j_5),(i_6,j_5),(i_5,j_6),$
$(i_6,j_6),(i_4,j_4)\}$
$\{(\forall\exists,\texttt{ct},\texttt{ct}),$
$(\exists,\texttt{ow},\texttt{ow}):C1,(\forall,\texttt{ow},\texttt{ow}):C3,$
$(\forall,\texttt{ow},\texttt{ow}):C8\}$
Concept 9:
$\{(i_5,j_5),(i_6,j_5),(i_5,j_6),$
$(i_6,j_6)\}$
$\{(\forall\exists,\texttt{ct},\texttt{ct}),(\forall\exists,\texttt{ow},\texttt{ow}):C0,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C1,(\forall\exists,\texttt{ow},\texttt{ow}):C2,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C3,(\forall\exists,\texttt{ow},\texttt{ow}):C7,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C8,(\forall\exists,\texttt{ow},\texttt{ow}):C12,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C13,(\forall\exists,\texttt{ow},\texttt{ow}):C14\}$
Concept 6:
$\{(i_5,j_5),(i_4,j_4),(i_5,j_5)\}$
$\{(\forall\exists,\texttt{st},\texttt{st}),(\forall\exists,\texttt{ct},\texttt{ct}),$
$(\forall\exists,\texttt{ow},\texttt{ow}):C1,(\forall\exists,\texttt{ow},\texttt{ow}):C3,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C8\}$
Concept11:
$\{(i_5,j_5),(i_6,j_6)\}$
$\{(\forall\exists,\texttt{st},\texttt{st}),(\forall\exists,\texttt{ct},\texttt{ct}),$
$(\forall\exists,\texttt{ow},\texttt{ow}):C0,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C1,(\forall\exists,\texttt{ow},\texttt{ow}):C2,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C3,(\forall\exists,\texttt{ow},\texttt{ow}):C7,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C8,(\forall\exists,\texttt{ow},\texttt{ow}):C12,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C13,(\forall\exists,\texttt{ow},\texttt{ow}):C14\}$
$\{\}$
$(\forall\exists,\texttt{st},\texttt{st}),(\forall\exists,\texttt{ct},\texttt{st}),$
$(\forall\exists,\texttt{st},\texttt{ct}),(\forall\exists,\texttt{ct},\texttt{ct}),$
$(\forall\exists,\texttt{ow},\texttt{ow}):C0,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C1,(\forall\exists,\texttt{ow},\texttt{ow}):C2,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C3,(\forall\exists,\texttt{ow},\texttt{ow}):C7,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C8,(\forall\exists,\texttt{ow},\texttt{ow}):C12,$
$(\forall\exists,\texttt{ow},\texttt{ow}):C13,(\forall\exists,\texttt{ow},\texttt{ow}):C14\}$

Figure 5.7 – Iteration 4: On the left the lattice of K_{c_1,c_3} and on the right the lattice of K_{c_2,c_4}

5.4 Concepts and link keys

The results of the RCA process are lattices composed of concepts containing as extent pairs of properties representing candidate link keys. However, as we can see in Figure 5.7, the extent are composed by duplicated object properties, each one associated to a different concept. Indeed, the RCA process propagates all the concepts of the lattice in relation. For instance, if we look at the concept 8, we can see both $(\forall\exists, \texttt{hm}, \texttt{hm}) : C4$ and $(\forall\exists, \texttt{hm}, \texttt{hm}) : C5$. As concept 4 is more specific than concept 5, we can remove $(\forall\exists, \texttt{hm}, \texttt{hm}) : C5$ without modifying the intent. More generally, to simplify candidate link key, we can simply remove the less specific pairs of object properties. Lattices of Figure 5.7 has been simplified and drawn in Figure 5.8.

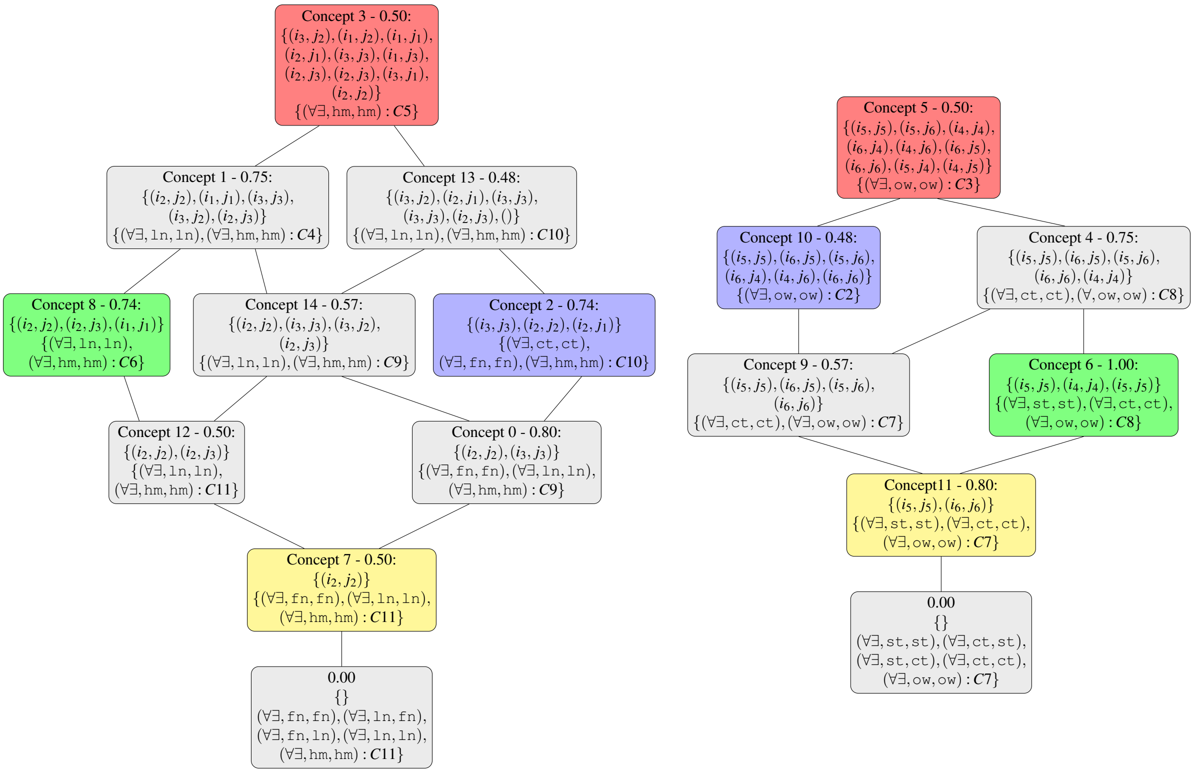

Concept 3 - 0.50:
$\{(i_3, j_2), (i_1, j_2), (i_1, j_1), (i_2, j_1), (i_3, j_3), (i_1, j_3), (i_2, j_3), (i_2, j_3), (i_3, j_1), (i_2, j_2)\}$
$\{(\forall\exists, \text{hm}, \text{hm}) : C5\}$
Concept 1 - 0.75:
$\{(i_2, j_2), (i_1, j_1), (i_3, j_3), (i_3, j_2), (i_2, j_3)\}$
$\{(\forall\exists, \texttt{ln}, \texttt{ln}), (\forall\exists, \text{hm}, \text{hm}) : C4\}$
Concept 13 - 0.48:
$\{(i_3, j_2), (i_2, j_1), (i_3, j_3), (i_3, j_3), (i_2, j_3), ()\}$
$\{(\forall\exists, \texttt{ln}, \texttt{ln}), (\forall\exists, \text{hm}, \text{hm}) : C10\}$
Concept 8 - 0.74:
$\{(i_2, j_2), (i_2, j_3), (i_1, j_1)\}$
$\{(\forall\exists, \texttt{ln}, \texttt{ln}), (\forall\exists, \text{hm}, \text{hm}) : C6\}$
Concept 14 - 0.57:
$\{(i_2, j_2), (i_3, j_3), (i_3, j_2), (i_2, j_3)\}$
$\{(\forall\exists, \texttt{ln}, \texttt{ln}), (\forall\exists, \text{hm}, \text{hm}) : C9\}$
Concept 2 - 0.74:
$\{(i_3, j_3), (i_2, j_2), (i_2, j_1)\}$
$\{(\forall\exists, \texttt{ct}, \texttt{ct}), (\forall\exists, \texttt{fn}, \texttt{fn}), (\forall\exists, \text{hm}, \text{hm}) : C10\}$
Concept 12 - 0.50:
$\{(i_2, j_2), (i_2, j_3)\}$
$\{(\forall\exists, \texttt{ln}, \texttt{ln}), (\forall\exists, \text{hm}, \text{hm}) : C11\}$
Concept 0 - 0.80:
$\{(i_2, j_2), (i_3, j_3)\}$
$\{(\forall\exists, \texttt{fn}, \texttt{fn}), (\forall\exists, \texttt{ln}, \texttt{ln}), (\forall\exists, \text{hm}, \text{hm}) : C9\}$
Concept 7 - 0.50:
$\{(i_2, j_2)\}$
$\{(\forall\exists, \texttt{fn}, \texttt{fn}), (\forall\exists, \texttt{ln}, \texttt{ln}), (\forall\exists, \text{hm}, \text{hm}) : C11\}$
0.00
$\{\}$
$(\forall\exists, \texttt{fn}, \texttt{fn}), (\forall\exists, \texttt{ln}, \texttt{fn}), (\forall\exists, \texttt{fn}, \texttt{ln}), (\forall\exists, \texttt{ln}, \texttt{ln}), (\forall\exists, \text{hm}, \text{hm}) : C11\}$
Concept 5 - 0.50:
$\{(i_5, j_5), (i_5, j_6), (i_4, j_4), (i_6, j_4), (i_4, j_6), (i_6, j_5), (i_6, j_6), (i_5, j_4), (i_4, j_5)\}$
$\{(\forall\exists, \texttt{ow}, \texttt{ow}) : C3\}$
Concept 10 - 0.48:
$\{(i_5, j_5), (i_6, j_5), (i_5, j_6), (i_6, j_4), (i_4, j_6), (i_6, j_6)\}$
$\{(\forall\exists, \texttt{ow}, \texttt{ow}) : C2\}$
Concept 4 - 0.75:
$\{(i_5, j_5), (i_6, j_5), (i_5, j_6), (i_6, j_6), (i_4, j_4)\}$
$\{(\forall\exists, \texttt{ct}, \texttt{ct}), (\forall, \texttt{ow}, \texttt{ow}) : C8\}$
Concept 9 - 0.57:
$\{(i_5, j_5), (i_6, j_5), (i_5, j_6), (i_6, j_6)\}$
$\{(\forall\exists, \texttt{ct}, \texttt{ct}), (\forall\exists, \texttt{ow}, \texttt{ow}) : C7\}$
Concept 6 - 1.00:
$\{(i_5, j_5), (i_4, j_4), (i_5, j_5)\}$
$\{(\forall\exists, \texttt{st}, \texttt{st}), (\forall\exists, \texttt{ct}, \texttt{ct}), (\forall\exists, \texttt{ow}, \texttt{ow}) : C8\}$
Concept11 - 0.80:
$\{(i_5, j_5), (i_6, j_6)\}$
$\{(\forall\exists, \texttt{st}, \texttt{st}), (\forall\exists, \texttt{ct}, \texttt{ct}), (\forall\exists, \texttt{ow}, \texttt{ow}) : C7\}$
0.00
$\{\}$
$(\forall\exists, \texttt{st}, \texttt{st}), (\forall\exists, \texttt{ct}, \texttt{st}), (\forall\exists, \texttt{st}, \texttt{ct}), (\forall\exists, \texttt{ct}, \texttt{ct}), (\forall\exists, \texttt{ow}, \texttt{ow}) : C7\}$

Figure 5.8 – Vector of link keys from the lattices of K_{c_1,c_3} and K_{c_2,c_4}

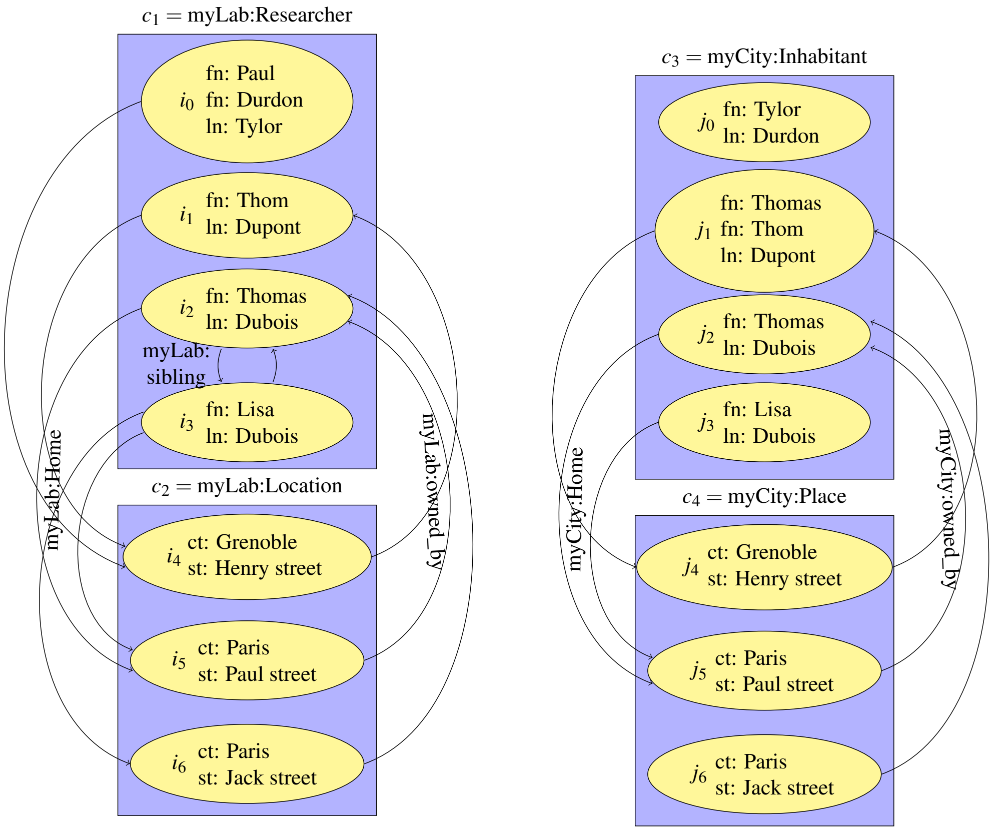
As one can notice, the selection of a link key among a set of candidate link keys cannot be performed individually for each lattice. We have to take into account the pairs of object properties and overall their associated concepts. For instance, it does not make sense to extract the link keys of the concepts 1 and 4 at the same time. On the first hand, the concept 1 is associated to the concept 4. Hence, the concept 4 seems to be compatible with the concept 1. On the other hand, the reciprocal statement is not true. The concept 4 is associated to the concept 8 which is more specific than the concept 1. Hence, if we choose as link key the extent of the concept 1, the links composing the intent of the concept 4 are not compatible.

The pair of compatible concepts have been colourised. Moreover, the f-measure of each link key is displayed beside the name of each concept. The process of selection of the best vector of link keys is simply to extract compatible concepts by applying the reasoning we made in the previous paragraph and by evaluating the sum of the f-measures of each vector. From this example, the compatible vectors are (C3, C5), (C8, C6), (C2, C10) and (C7, C11) with respectively a f-measure of 1.0, 1.74, 1.22 and 1.30. Hence, (C8, C6) is the vector of link keys that obtains the best f-measure and in consequence the link keys that allows the best linking. The two link keys are $\{(\forall\exists, \texttt{ln}, \texttt{ln}), (\forall\exists, \texttt{hm}, \texttt{hm})\}$ and $\{(\forall\exists, \texttt{st}, \texttt{st}), (\forall\exists, \texttt{ct}, \texttt{ct}), (\forall\exists, \texttt{ow}, \texttt{ow})\}$. In other word, identical researchers are identified by the fact that they have the same (and at least one) last names and the same (and at least one) homes. Identical homes are identified by the fact that they share the same (and at least one) cities and streets and the same (and at least one) owner.

Our proposed process explores all the candidate link keys in order to then build a vector made of the best compatible link keys. Contrary to what we remarked in the hierarchical approach of our FCA encoding process, the extracted link keys are optimal.

5.5 Noisy example

We propose here to make the data a little bit more complex by adding noise. As we saw, the previous example exhibits circular dependencies. However, none of the instances hold multiple values for a given properties. In order to expose this kind of situation, one instance has been added to each ontology. Moreover we also added multiple first name values as well as multiple home values to some of the existing instances. Hence, contrary to the previous examples, all the quantification will not apply for every pair of properties.



c_1 = myLab:Researcher
i_0 fn: Paul fn: Durdon ln: Tylor
i_1 fn: Thom ln: Dupont
i_2 fn: Thomas ln: Dubois
myLab: sibling
i_3 fn: Lisa ln: Dubois
c_2 = myLab:Location
i_4 ct: Grenoble st: Henry street
i_5 ct: Paris st: Paul street
i_6 ct: Paris st: Jack street
myLab:Home
myLab:owned_by
c_3 = myCity:Inhabitant
j_0 fn: Tylor ln: Durdon
j_1 fn: Thomas fn: Thom ln: Dupont
j_2 fn: Thomas ln: Dubois
j_3 fn: Lisa ln: Dubois
c_4 = myCity:Place
j_4 ct: Grenoble st: Henry street
j_5 ct: Paris st: Paul street
j_6 ct: Paris st: Jack street
myCity:Home
myCity:owned_by

Figure 5.9 – Case study 2: RDF graph of `myLab` on the left and `myCity` on the right

We run the RCA process on this example and obtain the two lattices of Figure 5.8. As before, the process terminates after 4 iterations. Adding noises to the data increases the size of the lattices and exposes more refined concepts. The vector of link keys resulting is (C12, C2). They correspond to the link keys $\{(\exists, \texttt{fn}, \texttt{fn}), (\forall\exists, \texttt{ln}, \texttt{ln}), (\forall\exists, \texttt{hm}, \texttt{hm})\}$ and $\{(\forall\exists, \texttt{ct}, \texttt{ct}), (\forall\exists, \texttt{st}, \texttt{st}), (\forall\exists, \texttt{ow}, \texttt{ow})\}$. Hence, with respect to the previous example, the addition of the first name has been necessary to discriminate the bad examples.

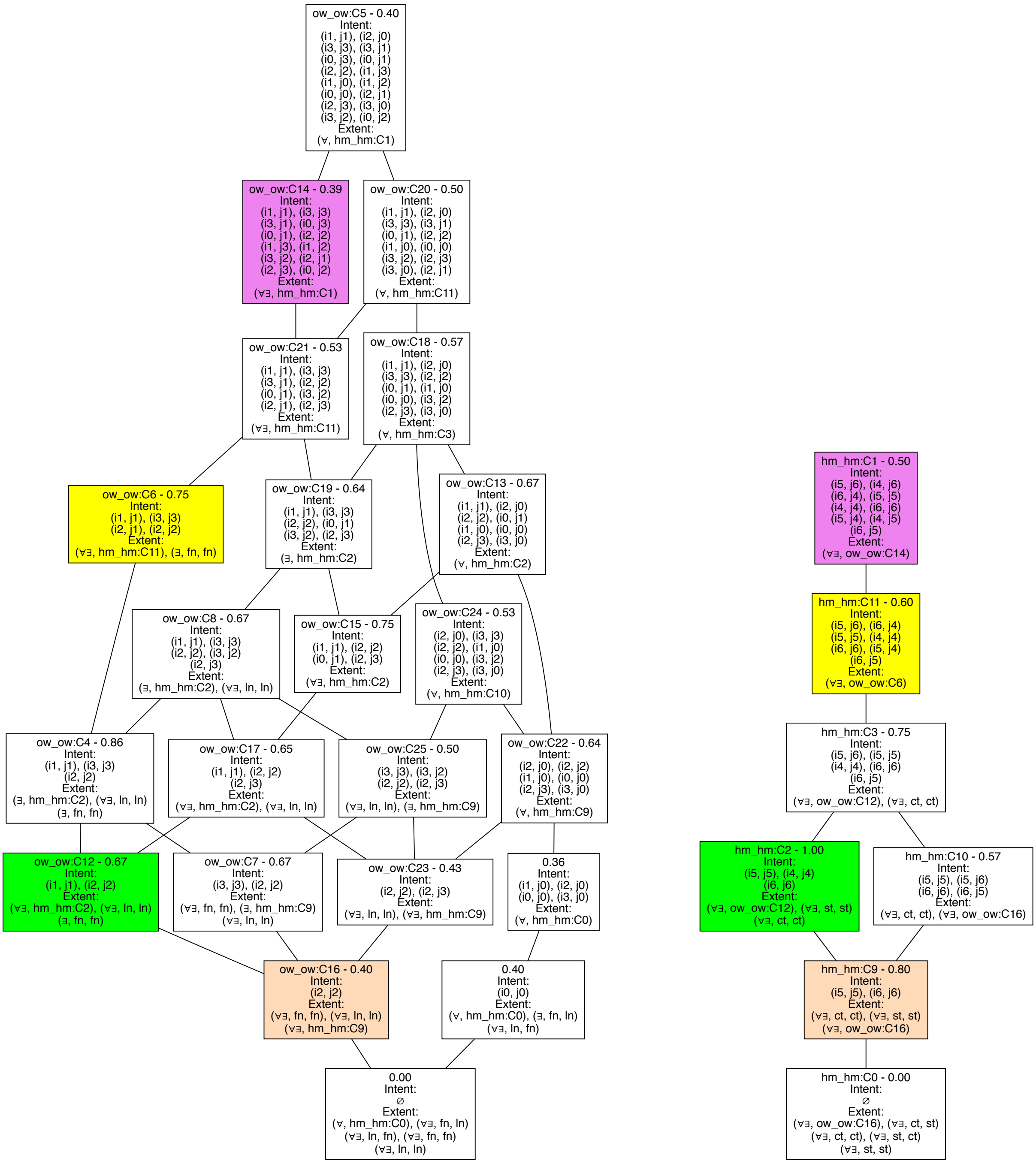


Figure 5.10 – Lattices resulting from the noisy example

5.6 Conclusion

In this chapter, we addressed the problem of circular dependencies. Moreover, we saw how to encode the link key extraction problem thanks to a relational context family. Then, we highlighted its behaviour on two examples and show the different steps. The proposed encoding is able to deal with both the multiplicity of values and the presence of object properties. From the built lattices we have been able to extract multiple valid combinations of candidate link keys and in consequence a set of link keys which is optimal according to the measures of selection. The process does not require any alignment information and supports circular dependencies.

In the next chapter, we confront our approach to more concrete real life example and evaluate it its performance. For that purpose, we compare it to other state of the art techniques. We start by presenting the implementation of the FCA and RCA encoding. Then we present the results obtained by the developed systems thanks to several experiments.

— 6 —
Implementation and experiments

This chapter presents both how we implemented the FCA and RCA encodings and the experiment we made. The first section details the implementation. It especially describes the different modules. The second section of this chapter is composed of experiments. We particularly present how to deal with the linking of instances about cities and arrondissement of data coming from INSEE and GeoNames.

6.1 Implementation

Our goal was to experiment the proposed encodings. However, we have not been able to find any open-source implementation of RCA. As the RCA algorithm uses FCA as foundation, we resolved to extend an available implementation of FCA in order to build our own RCA system.

For that purpose, we choose a FCA library [16] developed in Python and that internally uses the Norris algorithm [15]. We made this choice because this library proposes a very simple interface and advanced tools such as scaling operations and exploration algorithms. Our first attemps were to perform the FCA linking of two classes. For that purpose, we defined a textual input format and implemented a parsing module to run some tests. As a result, we have been able to reproduce the record linkage results of [3].

Our second try was to encode the FCA linking of two classes from INSEE-GeoNames. The original implementation of FCA used a non-sparse representation of the incidence tables. This characteristic caused memory issues due to the size of data set (see Section 6.2). In order to overcome the issues, we forked the original FCA module in order to opt for a sparse representation and consequently reduce the memory cost. The INSEE-GeoNames files are stored in the RDF format. So, we also had to integrate the RDFLib library to our implementation to be able to read them.

From this sparse implementation of FCA, we built the entire RCA process. We modified the lattice representations and defined all the RCA elements such as relational contexts and even scaled contexts. Due to efficiency issues, we implemented an ontology representation which provides access to the elements in constant time. We also added two modules in order to easily fetch the resulting link keys. Firstly, we added a module to export the lattices in a image format thanks to GraphViz. Secondly, in order to build the example of this document, the implementation has been fitted with a module exporting the tables in LaTeX. Finally, we implemented a module of selection of link keys in order to extract vectors of link keys from the produced lattices. Figure 6.1 illustrates the modules composing the RCA implementation.

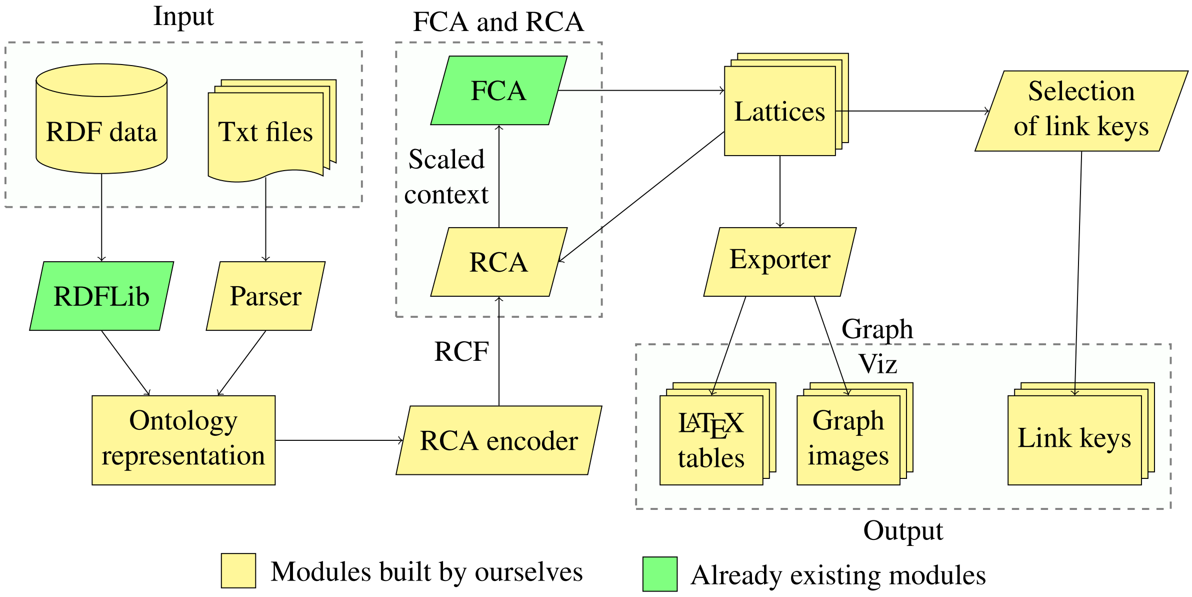


Figure 6.1 – Pipeline of the RCA implementation

6.2 Experiments

In this section, we present the experiments we made in order to evaluate the developed system. First, we present the compliance tests that allows us to test its correctness. Second, we aim at comparing link keys obtained thanks to our FCA encoding to another state of the art technique. Finally, we use the RCA encoding to link two ontologies that cannot be handled by other actual methods. As the ∀ quantification obtains very close measures to the ∀∃ one, we only use in the rest of this section the ∃ and ∀∃ quantifications.

Compliance test

In order to test our implementation, we run the examples provided in this document. All results presented in these examples have been computed by the developed system. Moreover, all these examples have been verified by hand and our implementation is able to provide the expected results.

The characteristics of the presented examples are specified in Table 6.1. The column #d-prop and #o-prop respectively correspond to the number of datatype properties and object properties. As one can notice, these examples contains only few instances. They do not constitute a real evaluation. That is why the next section evaluates the system by using a larger data set.

Data sets	#instances	#d-prop	#o-prop	#iteration	#concepts	Time
Figure 4.2	12	18	6	1	11	0.0s
Figure 5.2	12	24	12	4	17	0.0s
Figure 5.9	14	30	16	4	29	0.1s

Table 6.1 – Characteristics of the compliant experiments

INSEE-GeoNames with FCA

We aim at comparing the results of our FCA encoding to the method presented in [2]. Indeed, this study proposes an optimised algorithm allowing to extract link keys from two aligned classes. Hence, by using the same data, we are able to compare the resulting link keys. Moreover, as our actual implementation does not perform any optimisation and perfectly reflects the theoretical algorithm presented in this document, we aim at observing the limits of the latter.

The data set used is composed of two ontologies INSEE and GeoNames. They comprehend data about French cities and arrondissments. INSEE is organised in two classes ‘city’ and ‘arrondissement’ which respectively have 36697 and 343 instances. GeoNames is organised by a unique ‘feature’ class which has 36552 instances. INSEE cities and arrondissements have an object property `subdivisionDe` representing the region of the location. Similarly GeoNames features have two object properties `parentFeature` and `parentMD3` for the same purpose[1]. Figure 6.2 presents the RDFS graph of the two ontologies.

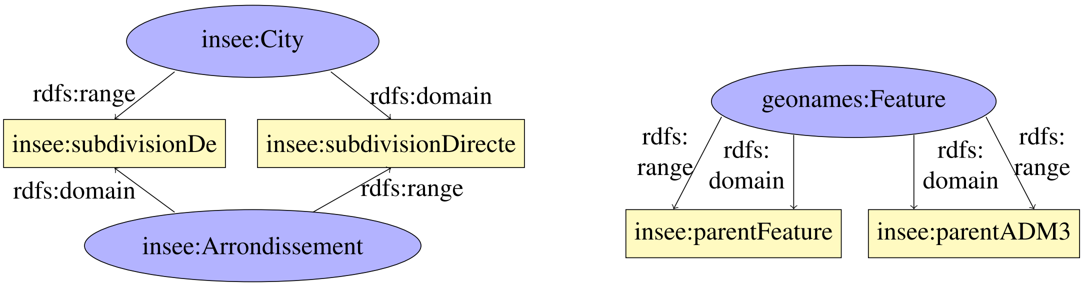


Figure 6.2 – RDFS graph of INSEE-GeoNames

For this experiment, we only use the city instances of the INSEE city class and of the GeoNames feature class. We used the exact same protocol as [2]. Both classes hold object properties. The related `sameAs` links has been provided in order to correctly evaluate the corresponding properties.

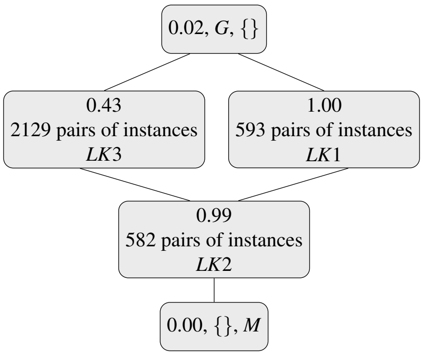
Because we encountered performance problems, we overcome them by starting with a reduced set of aligned instances and we then progressively increase the number of instances to look at the performance. The reduced sets are only composed aligned instances from the two classes.

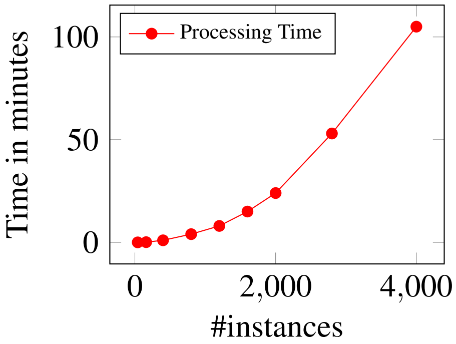
#instances	#d-prop	#o-prop	#concepts	Time	LK1 h-mean	LK2 h-mean
160	640	237	5	7s	1.00	0.99
400	1609	582	5	1min	1.00	0.99
1200	4796	1751	5	8min	1.00	0.99
2000	8065	2934	5	24min	1.00	0.99
2800	11270	4106	5	53min	0.99	0.99
4000	16167	5833	6	1h45min	0.99	0.99

Table 6.2 – Results of the FCA linking of INSEE-Geonames

[1] The presentation of such data sets with one feature class for everything and apparently identical properties `parentADM3` and `parentFeature` may seem strange. However, such thing happen in real world data sets and we did not modify them.

Table 6.2 contains the results of the experiments. We denote by LK_1 the link key $\{(\forall\exists\ ,\texttt{nom},\texttt{name})\}$, by $LK2$ the link key $\{(\forall\exists\ \texttt{nom},\texttt{name}),(\forall\exists,\texttt{subdivisionDe},\texttt{parentFeature}),(\forall\exists,\texttt{subdivisionDe},\texttt{parentADM3})\}$ and by $LK3$ the link key $\{(\forall\exists\ ,\texttt{subdivisionDe},\texttt{parentFeature}),(\forall\exists\ \texttt{subdivisionDe},\texttt{parentADM3})\}$.The lattice obtained by the experiment on 600 instances of each class is displayed below. We can see inside each node the obtained h-mean along with the number of pairs of linked instances and the candidate link key.





A first result is that we obtained the same link keys than the ones extracted in [2]. As one can notice, the size of the data does not influence the size of the lattice as well as the resulting link keys. It only influences the processing time what increases exponentially.

The extracted link keys are the same whatever the size of the sample. Hence it seems possible to bypass the issue by extracting randomly multiple samples and by performing the process on each one. The samples containing only few aligned instances would result in only bad candidate link keys. In contrast, the samples containing enough aligned instances would result on candidate link keys with good score. Hence, from these samples we would be able to extract the link keys and to then interlink the entire data set. A further work would be to evaluate the required proportion of the data sets in order to provide good results.

INSEE-GeoNames with RCA

The previous test was performed with FCA as in [2]. We now turn to test our RCA implementation. This time, we do not limit the process to only two aligned classes. On the opposite, we chose to extract a link key for both city and arrondissement. The city and arrondissement classes of INSEE as well the GeoNames feature class hold circular dependency. For each experiment, we selected the same number of aligned instances of each class. Moreover, we ensured that the extracted cities and arrondissements are in relations.

In spite of the ambiguity of the situation, we are able to understand from the results the alignment as well as the linking function. Contrary to [2] and to all the other existing methods, our RCA proposal does not require any information about the alignment. It extracts one link key for the pair of classes $\langle\texttt{insee:City},\texttt{geonames:Feature}\rangle$ denoted by $P1$ and a second link key for the pair $\langle\texttt{insee:Arrondissement},\texttt{geonames:Feature}\rangle$ denoted by $P2$.

We obtains the same vectors of link keys for each experiment: (LK1, LK1) and (LK2, LK1) where the first link key is used on P1 and the second on P2. Table 6.3 contains the results of the RCA process.

#instances	#d-prop	#o-prop	#concepts	#iteration	Time	*h* (LK1, LK1)	*h* (LK2, LK1)
80	320	78	10	2	3min	0.80	0.78
152	612	160	12	4	23min	0.79	0.77
312	1228	321	16	3	12h	0.79	0.76

Table 6.3 – Results of the RCA linking of INSEE-Geonames

Similarly to what has been observed in the FCA experiments, the size of the lattice does not increase with the size of the data set. It only increases the processing time. Figure 6.3 contains the lattices resulting from the RCA encoding with 152 instances.

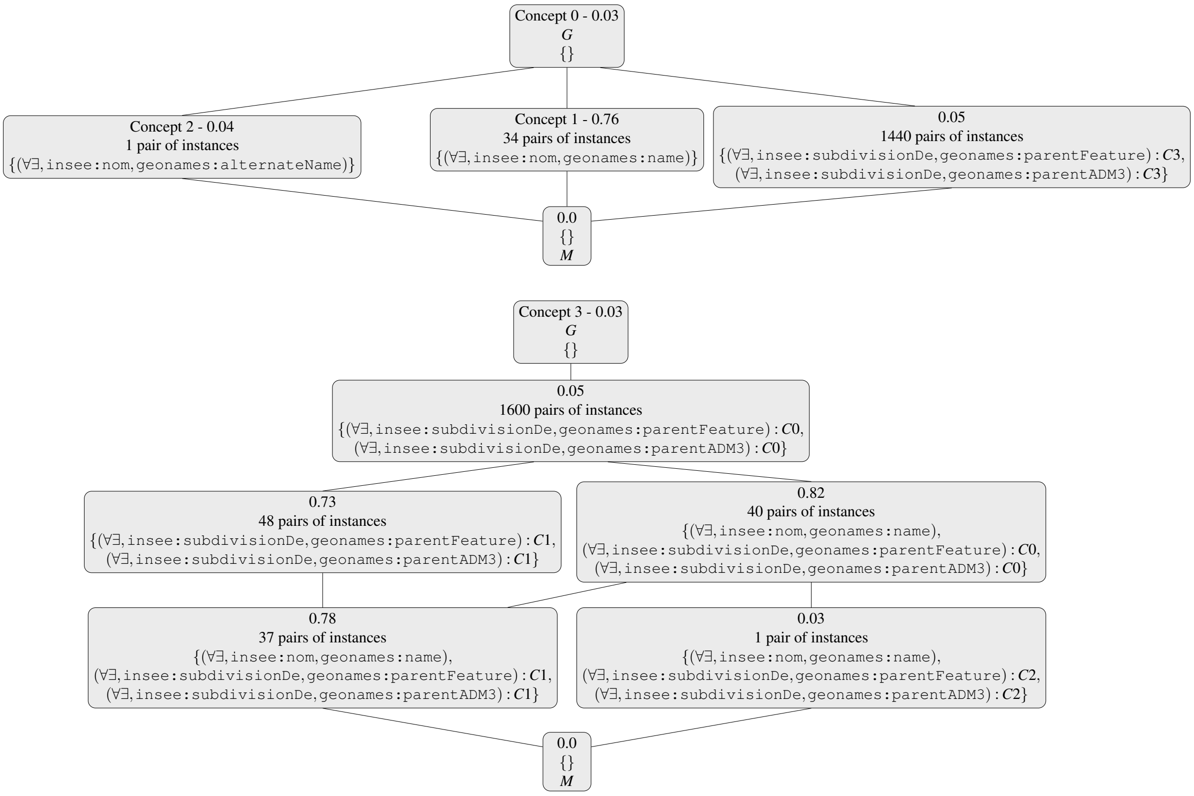

Concept 0 - 0.03
G
{}
Concept 2 - 0.04
1 pair of instances
{(∀∃, insee:nom, geonames:alternateName)}
Concept 1 - 0.76
34 pairs of instances
{(∀∃, insee:nom, geonames:name)}
0.05
1440 pairs of instances
{(∀∃, insee:subdivisionDe, geonames:parentFeature):C3,
(∀∃, insee:subdivisionDe, geonames:parentADM3):C3}
0.0
{}
M
Concept 3 - 0.03
G
{}
0.05
1600 pairs of instances
{(∀∃, insee:subdivisionDe, geonames:parentFeature):C0,
(∀∃, insee:subdivisionDe, geonames:parentADM3):C0}
0.73
48 pairs of instances
{(∀∃, insee:subdivisionDe, geonames:parentFeature):C1,
(∀∃, insee:subdivisionDe, geonames:parentADM3):C1}
0.82
40 pairs of instances
{(∀∃, insee:nom, geonames:name),
(∀∃, insee:subdivisionDe, geonames:parentFeature):C0,
(∀∃, insee:subdivisionDe, geonames:parentADM3):C0}
0.78
37 pairs of instances
{(∀∃, insee:nom, geonames:name),
(∀∃, insee:subdivisionDe, geonames:parentFeature):C1,
(∀∃, insee:subdivisionDe, geonames:parentADM3):C1}
0.03
1 pair of instances
{(∀∃, insee:nom, geonames:name),
(∀∃, insee:subdivisionDe, geonames:parentFeature):C2,
(∀∃, insee:subdivisionDe, geonames:parentADM3):C2}
0.0
{}
M

Figure 6.3 – Lattices from the RCA Insee-GeoNames with 152 instances: above the lattice of ⟨insee:arrondissement, geonames:feature⟩ and below the lattice of ⟨insee:city, geonames:feature⟩

6.3 Conclusion

In this chapter, we presented our implementation through the different modules composing it and their interactions. We then proceeded to a first evaluation of both our implementation and the two proposed encodings. The first experiment compared our FCA encoding to another state of the art technique. We obtained similar link keys than this previous work. The second experiment presented the linking of two ontologies by using the RCA encoding. Our method was able to go beyond the limits of the state of the art by interlinking them and overall by successfully dealing with the ambiguity raised by the lack of information about alignment of the classes. Actually, no other method is able to handle this kind of situation.

We also saw the limit of our implementation which could take advantage of further optimisations. [18] proposes a map-reduce version of Ganter's Algorithm that allows to extract formal concepts from a given formal context. Hence, similarly to our current implementation, a future work is to reuse this FCA algorithm and built at the top of it a map-reduce version of RCA algorithm.

The next chapter sums up the work done in this study.

— 7 —
Conclusion

The amount of RDF data sets through the web is growing as well as the number of systems requiring to interlink them. In this document, we presented a new approach to extract link keys. We especially extended the FCA encoding made in [3] for relational tables in order to interlink RDF data. This extension is not a trivial task due to two main problems: values for a given property can be multiple and values of properties are not necessarily datatypes but can be other objects of the graph.

We proposed an FCA encoding for pair of aligned classes. We have considered the EQ and IN quantifications in order to deal the multiplicity of values of the properties. Concerning object properties, we presented an hierarchical approach that allows to interlinking two ontologies when the pairs of aligned classes are known. This proposal is subject to two limitations. The first one is that the extracted link keys are not always the best. The second one is that the method cannot deal with data holding dependency cycles.

In order to go beyond the two limitations, we proposed to use RCA to extract link keys. We elaborated an encoding of the link key extraction problem that takes advantages of the relational aspect of RCA. This new encoding is able to deal with the multiplicity of values and with object properties. Moreover, the extracted link keys are always the best and circular dependencies are now supported. Another important point is that this encoding does not require any information about the alignment of the classes.

We provide an implementation of the methods in order to evaluate them. We have especially been able to reproduce results obtained in previous studies. By doing experiments on the INSEE-GeoNames data sets, we have been able to extract a link key for each pair of classes despite a total ambiguity raised by the feature class of GeoNames. Despite that we also observed the poor scalability of our implementation, none of the actual method is able to handle correctly such a situation.

Several developments can be undertaken in order to complete this study. On the one hand, our work provides an RCA encoding but reuses the classical RCA algorithm which forces us to construct all the concepts of all the lattices. Thanks to an evaluation of the concepts at each iteration, the trimming of certain concepts is conceivable. Similarly to a branch-and-bounds algorithm, a drastic reduction of the search is possible at each iteration. Moreover, a map-reduce version of the FCA has been proposed.

On the other hand, we saw that our method is able to link instances without any alignment information. Hence, from the extracted link keys, descriptions of classes can be induced in order to maximise the selection measures and to infer a description of the alignment. The initial aim of RCA is to induce class descriptions. Hence, it seems to be an adapted tool for the task.

Bibliography

[1] Manuel Atencia, Michel Chein, Madalina Croitoru, Jérôme David, Michel Leclère, Nathalie Pernelle, Fatiha Saïs, Francois Scharffe, and Danai Symeonidou. *Defining Key Semantics for the RDF Datasets: Experiments and Evaluations*, pages 65–78. Springer International Publishing, Cham, 2014.

[2] Manuel Atencia, Jérôme David, and Jérôme Euzenat. Data interlinking through robust linkkey extraction. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, ECAI'14, pages 15–20, Amsterdam, The Netherlands, The Netherlands, 2014. IOS Press.

[3] Manuel Atencia, Jérôme David, and Jérôme Euzenat. What can FCA do for database linkkey extraction? In *3rd ECAI workshop on What can FCA do for Artificial Intelligence? (FCA4AI)*, pages 85–92, Praha, Czech Republic, August 2014. No commercial editor. atencia2014d.

[4] Franz Baader, Ian Horrocks, and Ulrike Sattler. *Description Logics*, pages 3–28. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[5] Dan Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, World Wide Web Consortium, February 2004.

[6] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[7] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1997.

[8] Pascal Hitzler, Markus Krtzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 1st edition, 2009.

[9] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From shiq and rdf to owl: the making of a web ontology language. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1):7 – 26, 2003.

[10] M. Huchard, M. Rouane Hacene, C. Roume, and P. Valtchev. Relational concept discovery in structured datasets. *Annals of Mathematics and Artificial Intelligence*, 49(1):39–76, 2007.

[11] Marianne Huchard, Amedeo Napoli, Amine Mohamed Rouane Hacene, and Petko Valtchev. A gentle introduction to Relational Concept Analysis, Tutorial ICFCA 2011, May 2011.

[12] Sergei O. Kuznetsov and Sergei Obiedkov. Comparing performance of algorithms for generating concept lattices. *JOURNAL OF EXPERIMENTAL AND THEORETICAL ARTIFICIAL INTELLIGENCE*, 14:189–216, 2002.

[13] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3c recommendation, W3C, February 1999.

[14] Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. A survey of current link discovery frameworks. *Semantic Web*, (Preprint):1–18, 2015.

[15] Eugene M. Norris. An algorithm for computing the maximal rectangles in a binary relation. *Rev. Roum. Math. Pures et Appl.*, 23(2):243–250, 1978.

[16] Nikita Romashkin. Fca library. `https://github.com/ae-hse/fca`, 2011.

[17] François Scharffe, Zhengjie Fan, Alfio Ferrara, Houda Khrouf, and Andriy Nikolov. Methods for automated dataset interlinking. Deliverable 4.1, Datalift, 2011.

[18] Biao Xu, Ruairí de Fréin, Eric Robson, and Mícheál Ó Foghlú. Distributed formal concept analysis algorithms based on an iterative mapreduce framework. *CoRR*, abs/1210.2401, 2012.