*N° d'ordre :*

REPUBLIQUE ALGERIENNE DEMOCRATIQUE&POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR & DE LA RECHERCHE SCIENTIFIQUE


**UNIVERSITE DJILLALI LIABES**
**FACULTE DES SCIENCES EXACTES**
**SIDI BEL ABBES**

## THESE
## DE DOCTORAT EN SCIENCES

*Présentée par* **ATIG YAHIA**

*Spécialité :* *INFORMATIQUE*
*Option :* *Interoperability and integration of information systems on the Web*

*Intitulée*

# Découverte et Réparation des Alignements d'Ontologies dans le Web de Données Liées

*Soutenue le 26-10-2022*
*Devant le jury composé de :*

***Président :***

| | | | |
|---|---|---|---|
| **Mr. BOUKLI-Hacene** | **Sofiane** | **Prof. UDL-SBA** | **Président** |

***Examinateurs :***

| | | | |
|---|---|---|---|
| **Mr. BENSLIMANE** | **Sidi Mohamed** | **Prof. ESI-SBA** | **Examinateur** |
| **Mr. TOUMOUH** | **Adil** | **M.C.A UDL-SBA** | **Examinateur** |

***Directeur de thèse :***

| | | | |
|---|---|---|---|
| **Mr. BOUCHIHA** | **Djelloul** | **Prof. CUN Naâma** | **Directeur de thèse** |

***Co-Directeur de thèse :***

| | | | |
|---|---|---|---|
| **Mr. MALKI** | **Mimoun** | **Prof. ESI-SBA** | **Co-Directeur de thèse** |

*Année universitaire : 2022-2023*

بسم الله الرحمن الرحيم

الحمد لله و الصلاة و السلام على حبيبه محمد صلى الله عليه و سلم و على آله و صحبه و من والاه.

اللهم تقبل هذا العمل خالصا لوجهك الكريم.

إلى كل عالم أو متعلم أو محب لهما.

إلى أبي، أمي و محمد.. من يدفعون من الخلف.

إلى يونس، محمد و صارة.. من يجذبون من الأمام.

إلى زوجتي التي تمشي بجانبي.

# Acknowledgement

First of all, I would like to express my sincere recognition to Prof. Bouchiha Djelloul, Dr. Zahaf Ahmed and Prof. Malki Mimoun, my supervisors, for their encouragement and support during my research and even before.

My heartfelt thanks are intended for Prof. Boukli-Hacene Sofiane (UDL-SBA), who was willing to serve on my dissertation committee as president. Also, I thank Prof. Benslimane Sidi Mohamed (ESI-SBA) and Dr. Toumouh Adil (UDL-SBA), who served on the examination committee.

I would also like to thank all the researchers with whom I discussed at least one of the points concerning this dissertation. I especially thank the Professor Bellatreche Ladjel for his generosity, his advice and his warm welcome in his laboratory LIAS-ISAE at the ENSMA and the University of Poitiers. I also thank the Professor Ernesto Jimenez-Ruiz for his precious guidance and support to understand the alignment conservativity problem.

Last but not least, a special mention goes to my best friend M. Nouari Brahim and my colleagues Dr Khiati Nadri, Dr Bendida Aissam and Dr Braham Abdenour with whom I was in constant mutual discussion on problems related to our doctoral dissertations.

**Abstract.** The proliferation of ontology development has led to the appearance of ontology repositories to store and share ontologies and alignments. The usefulness of these repositories depends not only on ontologies quality, but also on alignments between them. Indeed, the evolution of ontologies following changes in their knowledge domains may affect and make obsolete the alignment between them. Thus, alignments must be evolved and maintained in order to keep up with the change in ontologies. While the main challenge for alignment evolution under ontology change methods is to maintain the alignment consistency after applying the change, the objective of this work is to take a step forward by considering the alignment evolution under ontological changes according to the conservativity principle. Two major contributions are brought by this dissertation: a methodology knowledge contribution and an in-depth analysis of the alignment evolution approaches. About the methodological contribution, the dissertation formally defines the alignment conservativity under the ontological change problem. This problem is then refined into two sub-problems. The first concerns the detection of conservativity violations, and here the dissertation proposes two patterns according to the type of ontological change. The second concerns the repair of the alignment, and there the Hitting set algorithm of diagnosis theory has been adapted for the alignment conservativity under the ontological change context. The literature analysis has led to classify the alignment evolution methods according to two categories. While the first calculates a new alignment from scratch by using ontology matching techniques, the second one reuses as much as possible the old alignment by adapting it to the ontological change. Based on this classification, this dissertation is positioned under the second works class to adapt the alignment according to new ontological requirements. Finally, the conducted experiment demonstrates on the one hand the practical applicability of the proposed approach, and on the other hand that ontology matching methods do not fit well for the alignment conservativity under the ontological change problem and suggests the current proposal as an add-on component to alignment evolution methods.

**Keywords.** Conservativity Principle Violations, Alignment Evolution, Ontological Change, Alignment Repair.

**Résumé.** La prolifération du développement d'ontologies a conduit à l'apparition d'entrepôts d'ontologies pour stocker et partager des ontologies et des alignements. L'utilité de ces entrepôts dépend non seulement de la qualité des ontologies, mais aussi des alignements entre elles. En effet, l'évolution des ontologies suite aux changements dans leurs domaines de connaissances peut affecter et rendre obsolète l'alignement entre elles. Par conséquent, les alignements doivent être évolués et maintenus pour suivre le changement dans les ontologies. Bien que le principal défi pour les méthodes d'évolution d'alignement suivant le changement ontologique est de maintenir la consistance de l'alignement après l'application du changement, l'objectif de ce travail est de faire un pas en avant en considérant l'évolution de l'alignement sous les changements ontologiques par rapport au principe de la conservativité. Deux contributions majeures sont apportées par cette thèse: un apport méthodologique et une analyse approfondie des approches d'évolution d'alignement. À propos de la contribution méthodologique, la thèse définit formellement le problème de la conservativité de l'alignement sous le changement ontologique. Ce problème est ensuite raffiné en deux sous-problèmes. Le premier concerne la détection des violations de la conservativité, et ici la thèse propose deux paternes selon le type de changement ontologique. Le deuxième concerne la réparation de l'alignement, et là l'algorithme Hitting Set de la théorie du diagnostic est adapté pour le contexte de la conservativité de l'alignement sous le changement ontologique. L'analyse de la littérature a abouti à classifier les méthodes d'évolution d'alignement selon deux catégories. Alors que la première calcule un nouvel alignement à partir de zéro en utilisant des techniques de matching d'ontologies, la deuxième catégorie réutilise autant que possible l'ancien alignement en l'adaptant au changement ontologique. Sur la base de cette classification, cette thèse se positionne sous la deuxième classe de travaux pour adapter l'alignement en fonction des nouvelles exigences ontologiques. Enfin, l'expérimentation démontre d'une part l'applicabilité pratique de l'approche proposée, et d'autre part que les méthodes de matching d'ontologies ne correspondent pas bien au problème de la conservativité de l'alignement sous le changement ontologique et suggère la proposition actuelle comme un composant plug-in aux méthodes d'évolution de l'alignement.

**Mots clés.** Violations du Principe de la Conservativité, Evolution de l'Alignement, Changement Ontologique, Réparation de l'Alignement.

**ملخص**. أدى انتشار تطور الأنطولوجيات إلى ظهور مستودعات الأنطولوجيا لتخزين ومشاركة الأنطولوجيات و المطابقات. لا تعتمد فائدة هذه المستودعات على جودة الأنطولوجيات فحسب، بل تعتمد أيضًا على جودة المطابقة بينها. في الواقع، يمكن أن يؤثر تطور الأنطولوجيات بسبب التغييرات في مجالات المعرفة على المطابقة بينها ويجعلها غير صالحة للإستعمال. لذلك، يجب تطوير المطابقات و صيانتها لمواكبة التغيير في الأنطولوجيا.

بينما التحدي الرئيسي لأساليب تطوير المطابقة بعد التغيير الأنطولوجي هو الحفاظ على اتساق المطابقة بعد تطبيق التغيير، فإن الهدف من هذا العمل هو اتخاذ خطوة إلى الأمام من خلال النظر في تطوير المطابقة في ظل التغييرات الأنطولوجية مع مراعاة مبدأ التحفظ.

تقدم هذه الأطروحة مساهمتين رئيسيتين: مساهمة منهجية وتحليل معمق لأساليب تطوير المطابقة. فيما يتعلق بالمساهمة المنهجية ، تُعرف الأطروحة رياضيًا مشكلة تحفظ المطابقة في ظل التغيير الأنطولوجي. هذه المشكلة تُقسم بعد ذلك إلى مشكلتين فرعيتين. تتعلق الأولى بكشف انتهاكات التحفظ، وهنا تقترح الأطروحة نمطين حسب نوع التغيير الأنطولوجي. و تتعلق الثانية بإصلاح المطابقة، وهنا يتم استعمال خوارزمية مجموعة الضرب لنظرية التشخيص و تكييفها لسياق المحافظة على المطابقة في ظل التغيير الأنطولوجي.

أدى تحليل الأدبيات إلى تصنيف طرق تطوير المطابقة وفقًا لفئتين. بينما يحسب الصنف الأول مطابقة جديدة إبتداءا من الصفر باستخدام تقنيات مطابقة الأنطولوجيات، يعيد الصنف الثاني استخدام المطابقة القديمة قدر الإمكان من خلال تكييفها مع التغيير الأنطولوجي. على أساس هذا التصنيف، تتموضع هذه الأطروحة ضمن الفئة الثانية من الأعمال لتكييف المطابقات وفقًا للمتطلبات الأنطولوجية الجديدة.

أخيرًا، توضح التجربة التي تم إجراؤها إمكانية التطبيق العملي للنهج المقترح من ناحية، ومن ناحية أخرى ، فإن طرق مطابقة الأنطولوجيات لا تتوافق بشكل جيد مع مشكلة تحفظ المطابقة في ظل التغيير الأنطولوجي وتقترح التجربة العمل الحالي كعنصر إضافي لطرق تطوير المطابقة.

**الكلمات المفتاحية :** انتهاكات مبدأ التحفظ، تطور المطابقة ، التغيير الأنطولوجي، إصلاح المطابقة.

# Contents

# List of figures

# List of tables



# List of abbreviations

- ASCII:  American Standard Code for Information Interchange
- CSV: Comma Separated Values
- DAG: Directed Acyclic Graph
- DL: Description logics
- HTML: HyperText Markup Language
- IRI: Internationalized Resource Identifiers
- JSON: JavaScript Object Notation
- JSON: JavaScript Object Notation
- KG: Knowledge Graph
- NLP : Natural Language Processing
- OBDA: Ontology-Based Data Access
- OWL: Web Ontology Language
- RDF: Resource Description Framework
- RDFS: Resource Description Framework Schema
- SHACL: Shapes Constraint Language
- SPARQL: SPARQL Protocol and RDF Query Language
- SPIN: SPARQL Inferencing Notation
- URI: Uniform Resource Identifier
- W3C: World Wide Web Consortium
- XML: eXtensible Markup Language

# Chapter 1: Introduction

## 1.1 Context

Ontology is defined as the conceptualization of objects recognized as existing in a domain of discourse with their properties and linking relationships (Neches et al., 1991). They play a very important role in computer applications which require for their functioning an overcoming of heterogeneity and diversity obstacles in semantics. The practical usefulness of ontologies is obvious with the emergence of many Semantic Web applications, allowing the Web in its current version to arise at the expense of its predecessors. For instance, ontology repositories such as OntoSelect (http://olp.dfki.de/ontoselect), DAML ontology library (http://www.daml.org/), Swoogle (http://swoogle.umbc.edu/2006/), Watson (http://kmi.open.ac.uk/technologies/name/watson/), and Schema.org (http://schema.org/) have successfully propagated within a large web community. They store, index, organize and share ontologies.

The problem is that given the same domain or related domains, it is possible that several ontologies developed simultaneously by different communities are available. Consequently, the comparison of two ontologies, passing from one to the other or integrating them becomes necessary through a discipline named Ontology Alignment. By ensuring semantic interoperability between ontologies (Motta & Sabou, 2006), the alignment between ontologies becomes therefore an indispensable task in many application domains (Euzenat & Shvaiko, 2013), for example to allow dynamically finding the appropriate ontology for a specific task. This became possible thanks to alignment repositories (such as : Bioportal (http://bioportal.bioontology.org), AgroPortal (http://agroportal.lirmm.fr/), Alignment server (http://alignapi.gforge.inria.fr/aserv.html), and RNASTAR (Widmann, et al., 2012) (https://github.com/alexdobin/STAR), which store, index, organize and share alignments.

## 1.2 Problem Statement

The need for semantic interoperability between ontologies does not make alignment faultless and impeccable, since mappings can lead to many undesirable logical consequences in the aligned ontologies and therefore the domains covered by these ontologies. Jiménez-Ruiz et al. (2011) proposed three principles to minimize the number of potentially unintended consequences, explicitly: (*i*) *Consistency principle*, the mappings should not lead to unsatisfiable classes in the integrated ontology, (*ii*) *Locality principle*, the mappings should link entities that have similar neighborhoods, (*iii*) *Conservativity principle*, the mappings should not introduce new semantic relationships between concepts from one of the input ontologies. It means that alignment should allow interaction between ontologies rather than providing a new description of the domain. Moreover, even if the alignment conservativity is well cared for during the calculation phase, or as a revision task just before its deployment, alignments such as ontologies are likely to be evolved throughout their life cycle (Euzenat et al., 2008; Stojanovic, 2004), and this evolution frequently degrades their qualities. As a result, alignments must be evolved and maintained in order to keep up with the change in input ontologies or to meet the demands of alignment applications and users.

Many methods have appeared to solve the problem of alignment evolution under ontology change (Dos Reis et al., 2013; Groß et al., 2013; Martins & Silva, 2009; Euzenat, 2015; Khattak et al., 2015; Zahaf & Malki, 2016). The main challenge for them is to maintain the alignment consistency after applying the change. An alignment is consistent if and only if the ontologies remain consistent even when used in conjunction with the alignment. The notion of consistency is approached by alignment evolution methods according to two different levels: structural consistency and logical consistency. The structural consistency ensures that the alignment obeys the constraints of its underlying representation structure (Martins & Silva, 2009). The logical consistency considers the semantics of the alignment, meaning that it does not introduce contradicting knowledge in ontologies (Euzenat, 2015; Zahaf & Malki, 2016).

## 1.3 Objectives

The objective of this work is to take a step forward by considering the alignment evolution according to the conservativity principle under ontological changes. In this

context, an alignment is conservative if the ontological change should not introduce new semantic relationships between concepts from one of the input ontologies. We consider these relationships as violations of the conservativity principle following ontological changes. We call such a situation as the *Alignment Conservativity Under Ontology Change* problem. This problem can be refined to include two other sub-problems, namely: the *Conservativity violations detection* and *Conservativity violations repair*.

**Problem 1 : Conservativity violations detection upon input ontologies evolution**

Knowledge is in frequent changes, which affects obviously the domains related ontologies. This evolutionary process does not stop there, but it goes beyond this to the linked alignments and affects their quality and usefulness for the connected applications (Euzenat, 2015). One of these situations is the unintended modification of the inferred knowledge in connected ontologies through these non-conserved alignments compared to that inferred in isolation. In other words, how to know if a change applied to an ontology does not affect the extracted inferences of other ontologies connected by an alignment and this evolved ontology. We call such a practice as *Conservativity Violations Detection upon input Ontologies Evolution*.

The affirmative answer to this question stipulates that alignment violates the conservativity principle. To face this position, we have two possible choices, either we calculate from scratch a new alignment for the new versions, which is not recommended on the fly considering the complexity of the alignment calculating process, or we adapt the original alignment according to the ontological changes. It should be noted that our objective here is not the detection of such a set of ontological changes but rather the impact of these changes on alignment correspondences. The question that arises here is how to distinguish between correspondences that do not cause conservativity violation in the original alignment, and those that do. The answer to this problem will then allow to restrict the intervention of a subsequent repair process on a subset in the original alignment rather than taking into account the whole of correspondences.

**Problem 2 : Conservativity violations repair upon input ontologies evolution**

Ontology evolution approaches cited in the literature (Stojanovic, 2004; Plessers, 2006; Klein, 2004) aim to produce evolution logs that store the implemented change. To exploit such a log, programs that jointly use different versions of an ontology must be able to

explicit the detected changes. Subsequently, the changes detection process within evolved ontologies can be essentially considered to enumerate the difference between their versions. This difference can then be used to identify and restrict the original alignment correspondences affected by the ontological change, i,e. correspondences which include one of the ontological entities involved in the changes. Therefore, when an ontological change causes conservativity principle violations, it is necessary to specify the subset of those correspondences directly responsible for these violations with respect to the considered change. Thereafter, the task of finding a relevant intervention on this subset in order to restore the conservativity of the original alignment is called a *Conservativity Violations Repair upon input Ontologies Evolution* problem.

## 1.4 Contributions

In this dissertation, we deal with the conservativity of alignment following the change in input ontologies. To the best of our knowledge, we are the first to study the problem of *Alignment Conservativity under Ontology Change* (Atig et al., 2022). We were able to make the following contributions:

1. We propose a method for detecting conservativity violations following the change in input ontologies. The method is based on two patterns depending on the type of ontological change. The former detects violations following the addition of new axioms to new ontology version, while the latter detects violations following the removal of axioms from this version. While existing methods of repairing conservativity violations (Solimando et al., 2016) only detect violations of the subsumption and equivalence types, our method detects violations of all types of axioms. We differentiate two possible situations in which the alignment can fall into conservativity violation depending on whether the violation appeared before or after the ontological change. In the context of the evolution of alignment under the ontology change, we are concerned with the second situation, that is, the violation of conservativity caused by the ontological change.

2. To deal with conservativity violations, we adopt a simple and efficient reparation approach, which consists in correcting alignment, while respecting its conservativity upon a change in its related ontologies. We adapt techniques from diagnosis theory (Reiter, 1987) to design this operation. A diagnosis is known to be

the minimal set of correspondences which intersects each minimal conflict set (Meilicke & Stuckenschmidt, 2007). The conflict set in turn represents a subset of correspondences responsible for each of the violations. The alignment repair process discards the diagnosis from the original alignment in order to restore its lost conservativity upon input ontologies evolution. The result of this revision is a repaired sub-alignment with respect to the conservativity principle.

3. We achieve an experiment on a relevant dataset adapted from the Ontology Alignment Evaluation Initiative (OAEI)[1] with a mixture of ontological changes applied to a set of tests. The experiment demonstrates the practical applicability of the proposed approach and shows the limits of ontology matching tools when dealing with alignment evolution problem with respect to conservativity principle. We notice that outputs of these tools suffer from this problem and propose our method as a complementary solution to cope. In fact, we do not consider our approach as a turnkey method to evolve an alignment, but rather as an additive component for this kind of approach, dealing with the conservativity violations upon ontological change.

4. We classify and examine related works according to two kinds of classes. The first calculates a new alignment from scratch by using ontology matching tools, while the second reuses as much as possible the old alignment by adapting it to the ontological change. Based on this classification, we position our proposal in the second works class to adapt the alignment according to the new ontological requirements.

5. The experience acquired from this dissertation allowed us to mention at the end a set of challenges of different nature. They can serve as future research questions into the alignment conservativity under ontology change problem.

[1] http://oaei.ontologymatching.org/

## 1.5 Dissertation Organization

We structure the remainder of this dissertation as follows:

Chapter 2 summarizes the basics concepts and definitions we will rely on along the paper. This section serves as background knowledge to understand the context of the alignment conservativity under ontology change problem. In Chapter 3 we introduce the problem statement by a motivating example, to well clarify our definition of the problem. After that, we show an examination of the conservativity principle problem studied in other related works. At the end, this analysis allows to position the problem studied in relation to all the work carried out previously. In Chapter 4, we unveil the patterns to detect conservativity violations following the source of the ontological change. Then, the alignment repair strategy is revealed. Chapter 5 presents the environment for implementing the proposal. Besides, this part also includes the conducted experimentation and discusses the obtained findings. Finally, Chapter 6 examines challenges of different nature representing open research issues, wraps up with concluding remarks and outlines future works.

# Chapter 2 : Ontology Alignment

## 2.1 Introduction

The current chapter is a crucial part for understanding the issues addressed by this dissertation. Theoretically then practically, it introduces ontologies and their expressiveness levels in section 2.2. Section 2.3 allows to explore the applications of ontologies in a set of famous Artificial Intelligence problems, which also pushes to discuss the usefulness of Ontology Alignment in these problems. Section 2.4 exposes in a concise but relevant way technologies of the Semantic Web which allow to build and exploit concrete ontologies. In Section 2.5, we clarify two important concepts in this work namely: ontology evolution and ontology versioning. In Section 2.6, we detail the notion of ontology alignment, which is inherently necessary to explain the context of the solutions proposed later in this work. The life cycle as well as the syntax and semantics are other additional points discussed in this section. Alignment evolution notion is examined at the end of this section. Finally, section 2.7 concludes the chapter.

## 2.2 Ontologies

In philosophy, ontologies are defined as a branch aiming to study the nature of things and their identities by answering questions about what things exist, with which attributes and in which groups can we categorize them. Indeed, this term is built starting from the Greek roots "*ontos*" which means what exists, the being, the existing, and "*logos*" which means the study or the speech, hence its translation by "The study of being" and by extension of "existence".

Born from the needs for knowledge representation, ontologies are transferred to Artificial Intelligence and be currently at the heart of Knowledge Engineering works. They aim to establish representations through which machines can manipulate the semantics of information. In this context, Neches et al. (1991) were the first to propose a brief and concise definition, namely: "*an ontology defines the basic terms and relations comprising*

*the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary*". This definition has subsequently undergone a series of improvements. Firstly, Gruber (1993) proposes : "*an ontology is an explicit specification of a conceptualization*". Then, with a slight modification, Borst (1997) reformulated the Gruber's definition by indicating that: "*Ontologies are defined as a formal specification of shared conceptualization*". Finally, Studer et al., (1998) have grouped these definitions together to specify that: "*An ontology is a formal, explicit specification of a shared conceptualization*". Other definitions have emerged, like Bernaras et al., (1996): "*An ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base*". Swartout et al,. (1996): "*A set of structured terms that describes some domain or topic. The idea is that an ontology provides a skeletal structure for a knowledge base*". A less concise definition has been proposed by Uschold & Gruninger (1996): "*Ontology is the term used to refer to the shared understanding of some domain of interest may be used as a unifying framework to solve problems ... An ontology necessarily entails or embodies some sort of world view with respect to a given domain. The world view is often conceived as a set of concepts (e.g. entities, attributes and processes), their definitions and their inter-relations; this is referred to as a conceptualization*".

This panoply of definitions and viewpoints had led Uschold & Gruninger (2004) to give a continuum of different kinds of ontologies shown in Figure 2.1. The spectrum of this figure increases from left (weak meaning) to right (strong meaning) the rate of expressivity of semantics as well as the formality of ontologies. A very simple meaning is expressed on the weak side, while arbitrary and complex meaning is expressed on the strong side. Hence, an ontology ranges from a simple set of terms with less or no explicit meaning to a simple notion of a taxonomy (knowledge with minimal hierarchy or structure), to a thesaurus (words and synonyms), to a conceptual model (with much complex knowledge) to a logical theory (which is very rich, complex, consistent, and a very significant knowledge).

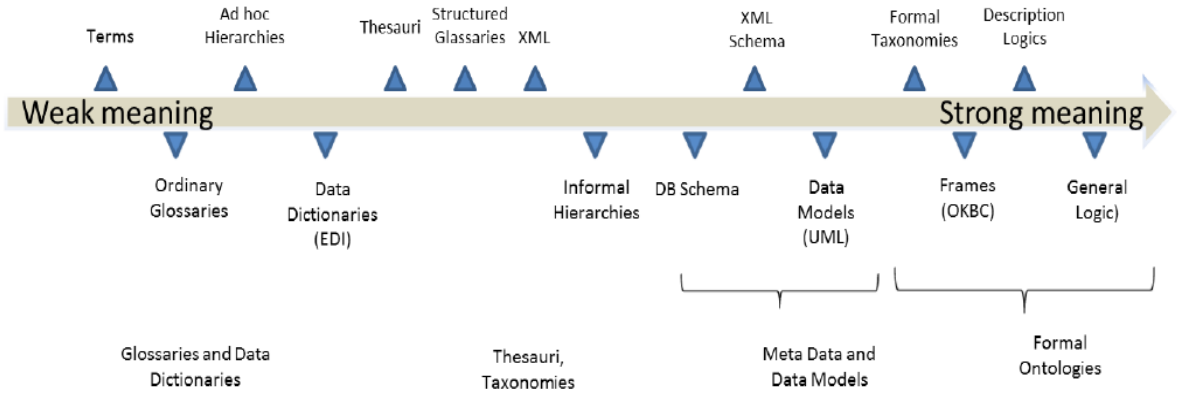


**Figure 2.1. Ontologies spectrum (Uschold & Gruninger, 2004)**

In this dissertation, we rely on the view of Kalfoglou & Schorlemmer (2003) and Grimm et al. (2011) to consider the ontology as a logical theory. This theory consists of a set of axioms that specify the intend interpretation of a vocabulary. In Definition 2.1, we formalize this vision.

**Definition 2.1** (Ontology as Logical Theory). An ontology $O$ is a couple $(S, A)$, with $A$ is a set of axioms that constraint the intended interpretation of a vocabulary $S$ also called signature $Sig(O) = C \cup P \cup R \cup I$ in a domain of discourse, where, $C$ represents the subset of vocabulary to designate concepts, $P$ is the subset of vocabulary to designate objects properties, $R$ is the subset of vocabulary to designate data properties, and $I$ is the subset of vocabulary to designate individuals.

The vocabulary of an ontology provides legal names for the entities appearing in the ontology, while axioms act as semantic constraints to define these entities. A signature and an instance of ontology using it is given in Example 1.

***Example 1.*** *Let S be a vocabulary composed by the following sets:*

$C$ = {*Person, Man, Woman, Child, Boy, Girl, Baby*}

$P$ = {*hasFather, hasMother, fatherOf, isBrotherOf*}

$R$ = {*hasName, hasAddress*}

$I$ = {*Yahia, Fatima, Younes, Mohamed, Sara*}

*Let A a set of axioms defined as follows:*

1. $Man \sqsubseteq Person$

2. $Woman \sqsubseteq Person$
3. $Child \sqsubseteq Person$
4. $Boy \sqsubseteq Child$
5. $Girl \sqsubseteq Child$
6. $Man \sqcap Woman \sqsubseteq \bot$
7. $Boy \sqcap Girl \sqsubseteq \bot$
8. $Boy \sqcup Girl \sqsubseteq Child$
9. $hasFather \equiv fatherOf^{-}$
10. *Man(Yahia)*
11. *Woman(Fatima)*
12. *Boy(Younes)*
13. *Boy(Mohamed)*
14. *Girl(Sara)*
15. *hasFather(Younes, Yahia)*
16. *fatherOf(Yahia, Younes)*
17. *hasMother(Sara, Fatima)*
18. *isBrotherOf(Younes, Mohamed)*

An instance of an ontology $O$ defined on top of signature $S$ is $O = (S, A)$.

An interpretation which satisfies all axioms of an ontology constitutes a model of that ontology. The model notion establishes a logical consequence relation between an ontology and axioms expressed in the language of this ontology.

**Definition 2.2** (Ontology Consequence). An axiom $\delta$ is a logical consequence of an ontology $O$ (noted $O \models \delta$) if and only if every model of $O$ satisfies $\delta$.

A set of ontological consequences is given in Example 2.

***Example 2.*** *Following Example 1, we can deduce the following ontological consequences:*

1. *Person(Yahia)*
2. *Person(Fatima)*
3. *Child(Younes)*
4. *Child(Mohamed)*
5. *Child(Sara)*

We denote by $Cn(O)=\{\delta|O\vDash \delta\}$ the closure set of logical consequences of an ontology $O$.

**Definition 2.3** (Ontology Inconsistency). An ontology $O$ is inconsistent if and only if $O$ has no model. Otherwise, it is consistent.

According to Hussain et al. (2011), the task of checking ontology inconsistencies returns in most cases to contradictory axioms entailment checking. Besides, if all models of an ontology lead to an unsatisfiable concept, this ontology is considered as incoherent (Flouris et al., 2006). A concept is unsatisfiable if no individual belongs to that concept for all interpretations. A concrete situation is shown in Example 3.

***Example 3.*** *Following Example 1, the following ontological consequences they make ontology inconsistent:*

1. *Girl(Younes)*
2. *Boy(Sara)*
3. *hasFather(Yahia, Younes)*

*On the other hand, the logical consequence isBrotherOf*(*Mohamed, Younes*), *does not make ontology inconsistent.*

Now that we have some idea of what an ontology is, let's see where they are being used.

## 2.3 Ontology Applications

Information systems have been the first ontologies users to help solving data integration issues. Applications of this domain are based on the common vocabulary of ontologies that is at one level of abstraction higher up than their conceptual data models (e.g, EER diagrams or UML Class Diagrams). Over the years, other applications for ontologies have been emerged.

### 2.3.1 Data system integration

#### 2.3.1.1 Integrating legacy systems

A very classic situation in schema-based data integration is the existence of multiple databases containing data on the same subject. Let's take for example a twinning project of two universities: each of the universities had its own database containing information on teachers, students, staff, etc. The objective here is therefore to merge the two databases into a single comprehensive one to manage data from both sides. One of the most promising

solutions is the ontology-based data integration. Although the problem of data integration is huge (Doan et al., 2012), we focus here only on the ontology-driven aspect. The use of one or more ontologies makes it possible to efficiently combine data or information from several heterogeneous sources (Wache et al., 2001). Data from multiple sources are characterized by multiple types of heterogeneity. Usually, this heterogeneity takes the following three forms:

- ***Syntactic heterogeneity:*** is due to the representation of data using different formats.
- ***Structural heterogeneity:*** is due to the difference in the original models or structures used to store data of the sources to integrate (Sheth, 1999).
- ***Semantic heterogeneity:*** is due to the lack of consensus on the exact meaning and interpretation of data values (Halevy, 2005).

Approaches using ontologies for data integration fall into two main classes (Wache et al., 2001). The first uses a single ontology as a global reference model in the system as presented in Figure 2.2. The Structured Knowledge Source Integration component of ResearchCyc[2] is a famous example of this class.

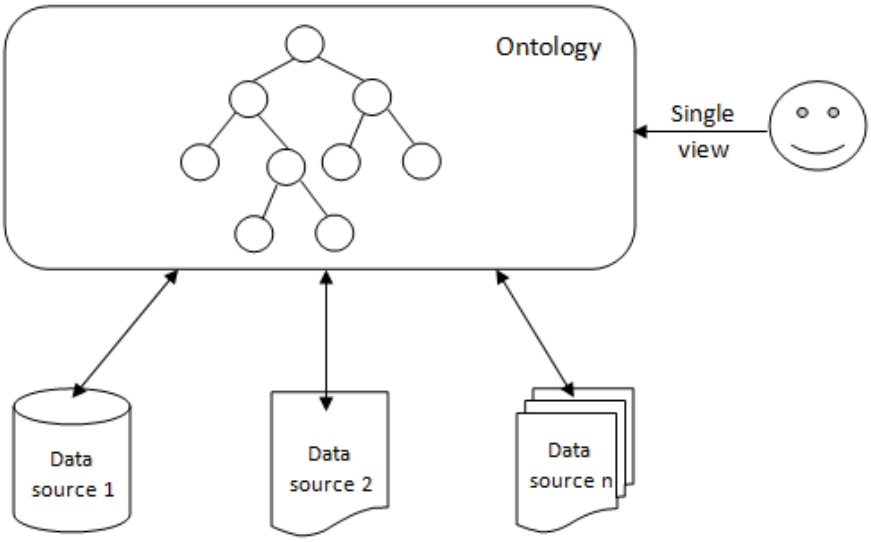


**Figure 2.2. A single ontology for data integration**

*Figure 2.2 shows that, using a single ontology as a global schema of different data sources (data source1, data source2,…source2), allows users to see multiple systems (with potentially obsolete languages) as one.*

[2] https://cyc.com/glossary/semantic-knowledge-source-integration/

The second class occurs when each data source is modeled by a private ontology as shown in Figure 2.3. In this case, the corresponding ontologies are used in combination for the integration. This requires the creation of mappings between these ontologies. The calculation of mappings between ontologies is known in the literature as the Ontology Alignment problem (see section 2.6). A possible third hybrid class serves to combine the first two classes (Goh, 1997). It uses an upper-level ontology to define the basic terms of a given domain, which allows integration by using multiple ontologies together with a common vocabulary between them.

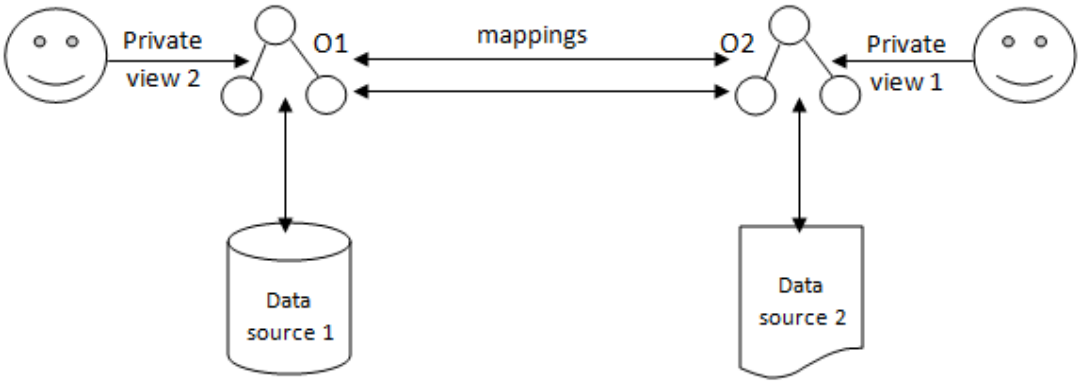


**Figure 2.3. Multiple ontologies for data integration**

*Figure 2.2 shows that, despite the differences in different/heterogeneous system schema, it is possible by using ontologies (O1 and O2) expressing different data sources (data source1 and data source2) and the alignment between them (set of mappings) to communicate the views at the conceptual level.*

### 2.3.1.2 Ontology-Based Data Access

Developed since the mid-2000s, Ontology-Based Data Access (OBDA) emerged as a data integration approach that allows querying different data sources through unified conceptual view of the application domain, expressed as an ontology (Poggi et al., 2008). The knowledge provided by ontologies allows users to ask a database without being aware of the underlying structure of the data. On the database administration side, OBDA not only avoids the need to know how data is stored, but also avoids the need to write very large queries (in some cases which may span pages), which makes it impossible to manage recurring queries. In addition, OBDA allows dissociate ontology from data through the use of declarative mappings, which greatly facilitates updating the data and its internal

structures. As an illustration, Figure 2.4 presents a simple OBDA architecture which includes an ontology, mappings and data sources.

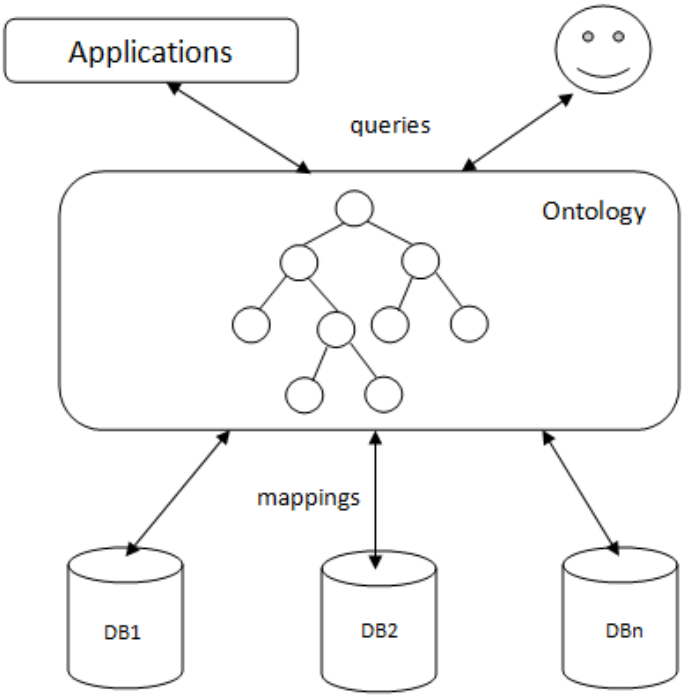


**Figure 2.4. OBDA architecture**

*Figure 2.4 shows three main components in an OBDA architecture: (i) an ontology, which enables unified conceptual view of managed information; (ii) databases ($DB_1$, $DB_2$,…$DB_n$), which are external, developed separately and potentially heterogeneous; and, (iii) a set of mappings, which play the role of intermediary between the ontology and databases. OBDA therefore allows users/applications to query different databases without having to know their private schemas or languages.*

The OBDA approach has seen significant success for many systems. As an example, one of the most promising real-world scenarios is the Slegger[3], an OBDA system designed for the data of Statoil (Equinor[4]), a Norwegian multinational oil and gas company which stores data in a large relational database (about 1500 tables and 1700 views). A typical use case of the Slegger effectiveness is presented in (Xiao et al., 2018). But not only, other use cases and industrial projects like Electronic Health Records (Rahimi et al., 2014), Manufacturing (Petersen et al., 2017), the Italian Ministry of Economy and Finance (Antonioli et al., 2014), Smart Cities projects (López et al., 2015) and the Statoil and

[3] http://slegger.gitlab.io/
[4] https://www.equinor.com/

Siemens optique project[5] have adopted OBDA systems like Mastro (Calvanese et al., 2011), Ultrawrap (Sequeda & Miranker, 2013), Morph (Priyatna et al., 2014), Stardog[6] and Ontop (Calvanese et al., 2016).

### 2.3.2 Semantic Web Search

The Web of documents has been successful because we humans are extremely good and flexible in processing data. We are able to read everything and acquire new knowledge. This success is clearly seen with the explosion in the amount of knowledge within billions of HTML pages to the point where we cannot comprehend or manage effectively. Indeed, despite the advancement of technologies in the field of Information Retrieval, traditional search engines remain limited to meet (complex and growing) user's requirements. Example 4, adapted from (Hogan et al., 2020) illustrates one situation of this limitation.

***Example 4.*** *Younes is a literature student. For his thesis work, he needs to find a list of Nobel Laureates in Literature who fought in a war, the year they were awarded the Nobel prize, and the name of the war(s) they fought in. From the list of such laureates on Wikipedia, he ends up manually checking the article for each laureate, looking for mentions of wars or conflicts, hoping he doesn't miss something. No one can deny that the web of documents has all the raw information that Younes needs. Using a traditional search engine, he can find a list of Nobel Laureates in Literature, and can check the Web to see if they have been involved in a war or not. Although the answer is available on the Web, it does not explicitly exist on a single web page that Younes can quickly find through a traditional search engine. This forces him to cross-reference different web pages to answer his own question.*

From Example 4, we notice that on one side, if the sought data is quite specific, there is little chance that it will be explicit on the Web: if there is not much demand for a precise information, then, there is less motivation for someone to make it explicit on a single web page. On the other side, traditional search engines were based mainly on the indexation of web pages and keywords search. Their results are displayed as lists of links to relevant pages on the Web. The user must read the text, look for the information that interests him and understand it. Hence the need to go to the next level to allow the best use of knowledge of the Web. A Web that involves semantics in the data of its resources and then offers the techniques to interrogate this data.

---

[5] http://optique-project.eu/
[6] https://www.stardog.com/

Although the solution was not so simple or instantaneous, because the research in this direction lasted more than two decades (Berners-Lee, 1998; Berners-Lee, 2001). Two avenues are already mature: (*i*) the way of publishing data by connecting them to a strict and precise meaning (see Section 2.4), and (*ii*) tools powerful enough to query these resources, what is called semantic search engines.

The goal of the semantic search engines, as one of the pillars of the Semantic Web is to push machines to collaborate to cross-reference data between them before assembling the relevant content into a single and concise web page of results for the user. This was adopted and implemented in recent years by some large organizations (Google, Bing, Yahoo, etc.).

Knowing that information on the Web are not described by a global schema over which queries can be expressed, also the difference in user languages, a semantic search engines has to rewrite the query with respect to available ontologies in order to use reasoning for providing answers (Euzenat & Shvaiko, 2013). Similar to the efforts of search engines to promote a common schema (eg: schema.org[7]), this solution can be effective only on the data encoded in a unique ontology. But, what if the searched data is scattered over several ontologies in the Web? In that case, it would be pertinent to match either the concerned ontologies or just the relevant parts in these ontologies: the parts where the sought concepts are found. In such situation, as shown in Figure 2.5, the usefulness of Ontology Matching solutions is very clear.

[7] http://schema.org

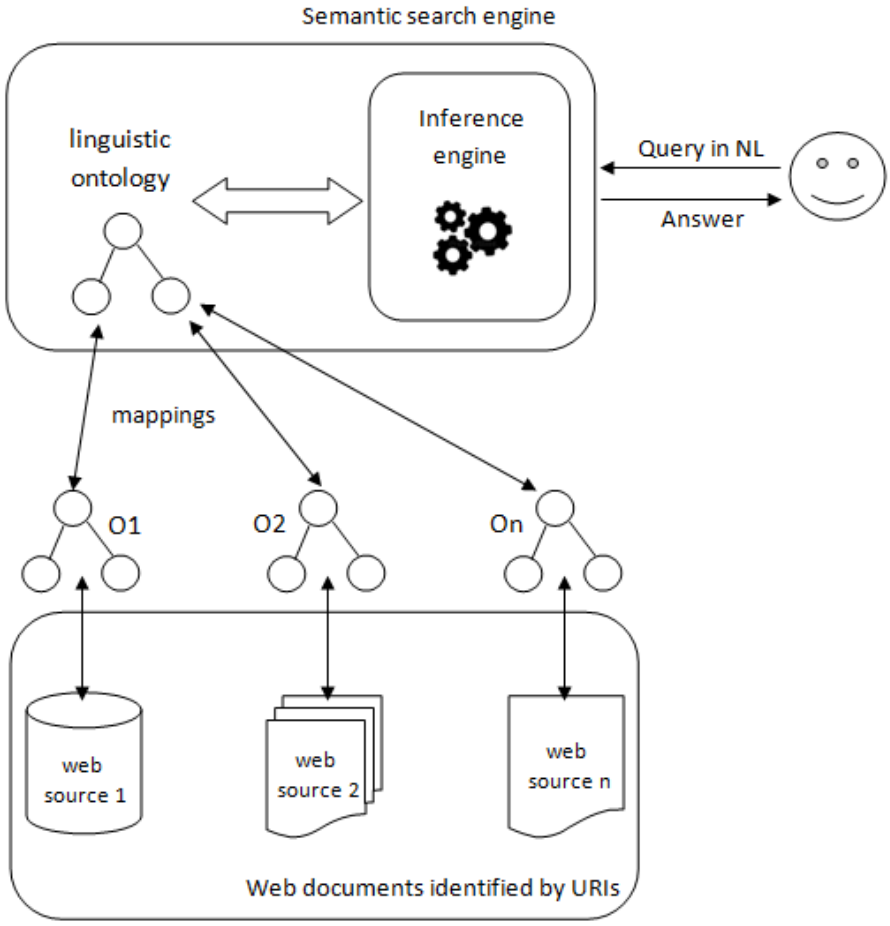


**Figure 2.5. Semantic search engine**

*In Figure 2.5, a semantic search engine is essentially made up of two parts: (i) linguistic ontology which plays the role of a translator between different languages of user queries and (ii) the inference engine to infer answers. The role of ontology matching is significant here since it allows interoperability between the ontology of the search engine and the ontologies ($O_1$, $O_2$,...$O_n$) under which different web data sources (web source$_1$, web source$_2$,... web source$_n$) are expressed.*

### 2.3.3 Natural Language Processing

Natural Language Processing (NLP) is one of the oldest fields of research that brings together artificial intelligence, computing and linguistics. This field has benefited from decades of intensive research efforts, as well as increasing computing power and availability of better corpora. Despite all this, NLP did not know its great maturity until after the Semantic Web era. Something that users of services like Google Translate, Siri[8]

[8] https://www.apple.com/siri/

can attest to. Indeed, ontologies provide an explicit and formal means for the interpretation, integration and sharing of data, helping to understand human natural language.

The existence of several natural languages and even of several ontologies for the same natural language triggers the need to look for the correspondences between these ontologies. This is where Ontology Matching techniques come in to propose solutions to interoperate semantics between the different linguistic ontologies.

If we go back to Example 4, it will therefore be possible now that we do the research using a single language (e.g., English) on different sources with private ontologies written in different languages. Something that further enhances the potential to exploit more knowledge on the Web.

### 2.3.4 Linked Data

The principle of the Web of Data is to publish data instead of full web pages. For this purpose, Berners-Lee (2009) and Heath & Bizer (2011) have set four principles for publishing data on the web of data: (*i*) Resources are identified by URIs (see Section 2.4.1); (*ii*) URIs are dereferenceable; (*iii*) when an URI is dereferenced, a description of the identified resource should be returned, ideally adapted through content negotiation; and, (*iv*) published web datasets must contain links to other web datasets.

The most adequate way to respect the web of data principles and therefore make the linked data more usable is to use the Semantic Web technologies (Euzenat & Shvaiko, 2013), namely, URIs for identifying resources, RDF for describing them, OWL for defining the used vocabularies and SPARQL for accessing data. We give more details about these technologies in Section 2.3.

But how can we measure this usability? In 2010, Tim Berners-Lee suggested the "five stars rating" for Linked Data, starting with one star, with data getting more stars when proprietary formats are removed and links are added, explicitly :

| | |
|---|---|
| ★ | Publish data on the Web using any format, (e.g. an image scan of a table); |
| ★★ | Use machine-readable structured data (e.g. Excel instead of image scan of a table); |

| | |
|---|---|
| ★★★ | Use non-proprietary formats, (e.g., CSV instead of Excel) to allow access to raw data; |
| ★★★★ | Use open standards from W3C (RDF and SPARQL) to identify things, so that people can point at your data; |
| ★★★★★ | All the above, plus: Link your data to other people's data to provide context. |

The web of documents allowed reaching up to the first three stars, which has already permitted some data reuse. However, the objective of linked data, and by extension of the web of data, is to have data more easily discoverable and interoperable, which makes it necessary to arrive at the fourth and fifth stars. This is possible thanks to RDF for publishing data, OWL for describing vocabularies, SPARQL for providing access points and a set of links between datasets. The most popular example here is the Linked Open Data[9], a project to link data which is released under an open license, hence, nothing prevents their free reuse. The dataset contained as of May 2020, 1255 datasets with 16174 links.

An important problem in linked data is being able to establish links between datasets. This is achieved by seeking in different datasets, entities representing the same resource, and linking their URIs using the owl:sameAs predicate. At first glance, this may be easy through existing methods such as record linkage[10] in databases (Fellegi & Sunter, 1969; Elfeky et al., 2002) or entity identification[11] (Lim et al., 1993) which aim at identifying multiple representations of the same entity within a set of entities. The problem with such solutions is that they are usually performed in a single database, that is, the same schema is used to describe all entities. However, in an open environment such as the Semantic Web, data are expressed using multiple and heterogeneous schemas or ontologies. Therefore, ontology matching is proposed as an adequate solution to remedy this problem, and several approaches have emerged (Scharffe & Euzenat, 2011; Salvadori et al., 2017). Figure 2.6

[9] https://lod-cloud.net/

[10] Record linkage is the process of finding records in a set of data that refer to the same entity in different data sources (e.g. different databases). It is needed when joining different datasets based on entities which may or may not share a common identifier.

[11] Entity identification (also known as (named) entity extraction, chunking, and entity recognition) is an information extraction sub-task which aims to find and classify named entities mentioned in different databases into predefined categories.

(inspired from (Scharffe & Euzenat, 2011)) is a use case of ontology matching in the context of web of data.

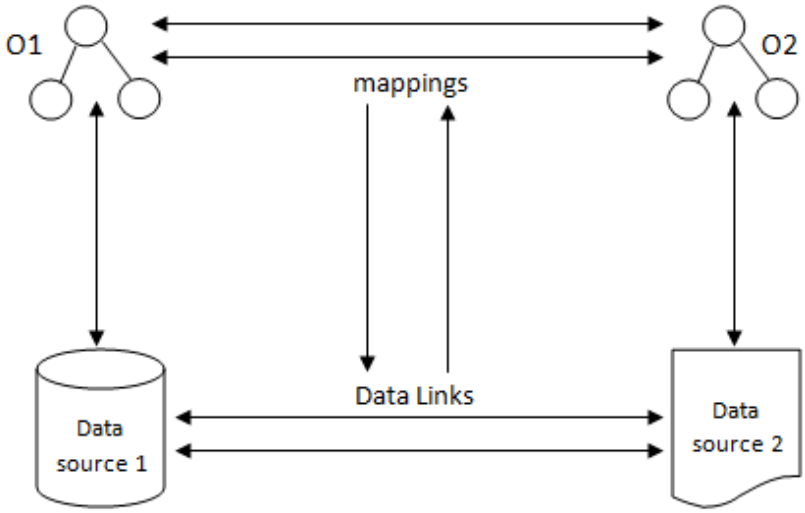


**Figure 2.6. Data interlinking and ontology matching**

*Figure 2.6 shows a simple scenario in the Web of data context, where ontology matching is useful to help generating links between different datasets: (i) a set of mappings is calculated between the ontologies ($O_1$ and $O_2$) which express the concerned datasets (data source$_1$ and data source$_2$); (ii) a set of links (owl:sameAs relations) is generated using those mappings for interlinking the datasets; and (iii) these links can also be used for improving matching.*

### 2.3.5 Knowledge Graphs

The effervescence of data available on the web has drawn with it a panoply of types, formats and an infinite number of data sources. On the opposite side, and since every action has a reaction, as Newton's laws of motion claim, an arsenal of techniques and tools for the extraction, storage, processing and analysis of such data have emerged in recent years to enable this variety to be exploited and managed as well as possible. One of the most important structures to do this is called a Knowledge Graph (KG). Knowledge Graphs are not new in the literal sense of the word since they are lightweight versions of semantic networks (Lehmann, 1992), but restored by their massive use by industry (critical to the functions of intelligent virtual assistants such as Google Assistant[12], Siri[13] and

[12] https://assistant.google.com/

Alexa[14].). Indeed, since the 1970s several areas have used KGs (artificial intelligence (Danilo et al., 2020), linked data, big data, Open Knowledge Network, and deep learning(Gao et al., 2020)) and with the advent of the Semantic Web and its related fields of research, they have proven to be of particular importance.

An appropriate methodology for the creation of a KG depends on a set of factors, among them: the envisaged purposes and applications, the field, the actors involved, the available data sources, etc. These last can range from unstructured plain text to structured formats (including the whole range between the two). Such a process should be flexible and the result is an initial core which can be gradually enriched from other sources as needed. In this context, two examples are proposed: either an Agile methodology (Hunt & Thomas, 2003) or "pay-as-you-go" (Sequeda et al., 2019).

The KGs creation and enrichment methods depend on the availability of data within the sources to be mined. However, there is no guarantee that this data was not incomplete, inconsistent or imprecise, especially when it comes from several sources. Therefore, a step for evaluating the quality of the resulting knowledge graph is crucial. Here too, the definition of quality may be different depending on the objectives targeted during the initial creation and enrichment of a knowledge graph from external sources and according to the purposes and applications envisaged, as well as the domain and the aimed context.

Despite the differences in ontology definitions, views and uses, they all agree that constructing concrete ontologies requires both a study of human knowledge and the definition of representation languages, as well as the creation of systems to manipulate them. Human knowledge study is expressed by the first stages of knowledge engineering techniques: knowledge acquisition and knowledge validation. Once this knowledge is acquired and validated, it should be represented in a language that makes possible its use by knowledge systems. Due to lack of logic-based semantics in the first proposals languages for representing ontologies such as semantic networks[15] or Cycl (Lenat & Guha,

---

[13] https://www.apple.com/siri/

[14] Alexa is a voice-activated virtual assistant. It can play audio, control a smart home, answer questions and use preferred services to keep the user organized, informed, safe, connected and entertained. As an Amazon product, she is also a personal shopper.

[15] A semantic network, also called frame network is a form used for knowledge representation. It is a knowledge base including semantic relations between concepts in a network.

1989), KIF (Genesereth et al., 1992), Ontololingua (Farquhar et al., 1997), Flogic (Kifer et al., 1995) which are based on *frames* (Minsky, 1975) combined with first order logic, Description Logic was proposed to surmount this insufficiency (Baader et al., 2003). Classic (Patel-Schneider et al., 1991) and LOOM (MacGregor, 1999) are two examples based on description logics. The languages of the Semantic Web were designed based on three paradigms: (*i*) RDF (Lassila & Swick, 1999) itself based on semantic networks to describe web resources, (*ii*) RDFS (Brickley et al., 2004) which adds frame primitives to RDF to organize web metadata and (*iii*) OWL (Dean et al., 2004) which is built on RDF(S). OWL includes some functionality of frames and other description logic to more explicitly specify the semantics of the vocabulary. We cover in the next section the RDF Data Model, the RDFS Data Schema Model and the Web Ontology Language OWL.

## 2.4 Ontology Languages

Following different abstract models, several formats already exist on the Web for publishing data. We can cite for example, CSV, JSON and XML which are the most popular but not the only ones. While CSV represents tables, JSON and XML both represent trees. The choice between these formats is based, on the one hand, on the data types to be manipulated, and on the other hand, on the potential consumer applications of this data. This choice gets complicated when we have to integrate data from different sources on the Web that use different formats with different models. A non-trivial solution, which may need human expertise, proposes to convert the source format to target format and vice versa (CSV to XML, XML to CSV, or JSON to XML…). Another solution is to use tools (e.g., relational databases) that support multiple formats. There too, the task is not easy, since it will be required to design complex queries supporting several formats and models. Thus, integration does not actually take place until the query time. The ideal solution is to consent on a standard format for publishing and exchanging data on the Web. This excludes the need to integrate heterogeneous existing data formats and makes it easy to map them to a single standard. This standard serves as a basis for the new version of the Web: the Semantic Web. Thereby, the World Wide Web Consortium[16] (W3C) proposes Resource Description Framework (RDF) to play the role of a standard data model. Indeed,

[16] https://www.w3.org/

with its graph-based data model, RDF is more flexible and easier to integrate than trees or tables.

### 2.4.1 Resource Description Framework (RDF)

As indicated by its name, the Resource Description Framework (RDF) is a standard framework for describing knowledge (data as well as metadata) using fixed structure expressions. In fact, the fundamental structure of any expression in RDF is a collection of triplets, each one composed of a *subject*, a *predicate* and an *object*. While the subject designates the resource itself and the object designates either a value or a subject in another triplet, the predicate designates features or aspects of this resource, and is used to express a relation between the subject and the object. These three components form what is called an RDF graph. As shown in Figure 2.7, this can be concretized by a Directed Acyclic Graph (DAG) composed of nodes and labeled directed arcs connecting pairs of nodes, in which each RDF triplet is represented by a node(subject)-arc(predicate)-node(object) link.

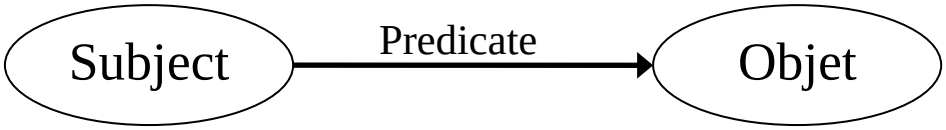


**Figure 2.7. RDF Triplet**

An RDF node can be either:

- *IRIs*[17] *(Internationalized Resource Identifiers):* RDF was originally based on URIs (Uniform Resource Identifiers) (Berners-Lee et al., 2005) which are limited to a subset of ASCII characters with no diacritics. This requires the need to the *percent encoding*, such that a substring "étudiant" becomes the significantly less readable "%C3%A9tudiant". Therefore, the IRI was proposed as a generalization of URIs to allow such characters, permitting substrings such as "étudiant". The use of IRIs was introduced in RDF 1.1. To be concise, the example in Figure 2.8 shows the structure of a typical IRI.

[17] https://www.w3.org/International/O-URL-and-ident.html

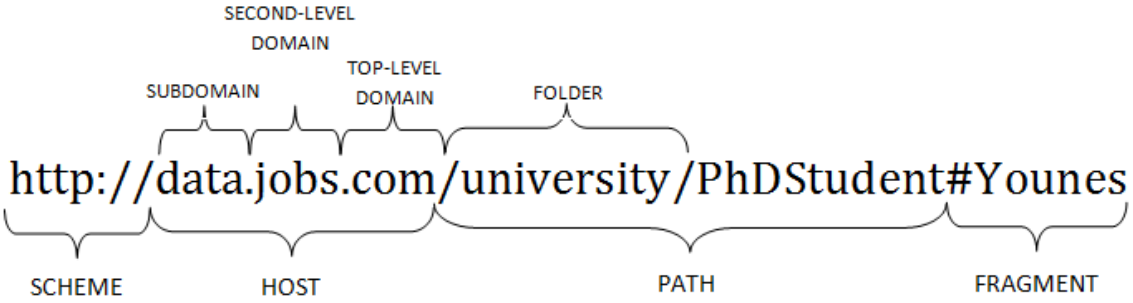


**Figure 2.8 Structure of a typical IRI**

- *Literals*[18]*:* lexical strings to represent numbers, booleans, dates, etc., or
- *Blank nodes:* Blank nodes may be given as document-local identifier called a blank node identifier. Predicates are IRIs and can be interpreted as either a relationship between the two nodes or as defining an attribute value (object node) for some subject node.

Since RDF is an abstract model, it requires serialization (i.e., file formats). Therefore, several serialization formats are available to write RDF triplets. We can cite:

- **RDF/XML**[19], an XML-based syntax that was the first standard format for serializing RDF.
- **Turtle**[20], a compact, human-friendly format.
- **N-Triples**[21], a very simple, easy-to-parse, line-based format that is not as compact as Turtle.
- **N-Quads**[22], a superset of N-Triples, for serializing multiple RDF graphs.
- **JSON-LD**[23], a JSON-based serialization.

[18] An RDF literal may include three parts (Cyganiak et al., 2014): (*i*) a lexical form written as a Unicode string, (*ii*) a datatype IRI that identifies the type of literal, and (*iii*) a language tag indicating the natural language of the text.
[19] https://www.w3.org/TR/rdf-syntax-grammar/
[20] https://www.w3.org/TR/turtle/
[21] https://www.w3.org/TR/n-triples/
[22] https://www.w3.org/TR/n-quads/
[23] https://www.w3.org/TR/json-ld/

- **N3** or **Notation3**, a non-standard serialization that is very similar to Turtle, but has some additional features, such as the ability to define inference rules.
- **RDF/JSON**[24], an alternative syntax for expressing RDF triples using a simple JSON notation.

RDF/XML serialization is sometimes confused with the RDF data model (in many cases RDF/XML is referred to simply as RDF) because it was historically the first standard RDF serialization format of the W3C. We do not go here into RDF details presented by the W3C specifications. Instead, we illustrate the RDF/XML serialization in Example 5.

***Example 5.*** *The triplet (www.jobs.com#PhDStudent, hasJob, Younes) is translated by:*

```
<rdf:Description about="http://www.jobs.com#PhDStudent">
      <hasJob>"Younes"</hasJob>
</rdf:Description>
```

*Example 5 illustrates in a simple way that "Younes" is a "PhDStudent". On the other hand, the "hasJob" property only makes sense to describe the "http://www.jobs.com#PhDStudent" resource in a well-defined context. Firstly, the reader (user) must be human; secondly, this reader understands English; and finally, the information transmitted by the RDF {http://www.jobs.com#PhDStudent, hasJob, Younes} triplet is quite simple to designate that "Younes has a job".*

### 2.4.2 Resource Description Framework Schema (RDFS)

The primary aim of the Semantic Web is to give sense of the information stored as RDF triples, to provide a vocabulary defining the meaning of properties within these triples, such as the "hasJob" property of Example 5, as well as its type, its values, etc. This is where the RDF-Schema[25] (RDFS) comes in. Another standard proposed by W3C to define data semantics. Its role is to allow the creation of metadata vocabularies through the definition of *classes* and *properties*.

In the same way as RDF above, we do not detail here all that the W3C has standardized as syntax for RDFS. By cons, we present in Example 6 an overview of the RDFS philosophy.

[24] https://www.w3.org/TR/rdf-json/
[25] https://www.w3.org/TR/rdf-schema/

***Example 6.*** *A set of RDF triples is organized in RDF-Schema as follows:*

```
<rdfs:Class rdf:ID='Lecturer'/>
<rdfs:Class rdf:ID='PhDDirector'>
        <rdfs:subClassOf rdf:resource='#Lecturer'/>
</rdfs:Class>
<rdfs:Class rdf:ID='Researcher'/>
<rdfs:Class rdf:ID='PhDStudent'>
        <rdfs:subClassOf rdf:resource='#Researcher'/>
</rdfs:Class>
<rdf:Property rdf:ID='Supervisor'>
        <rdfs:domain rdf:resource='#PhDDirector'/>
        <rdfs:range rdf:resource='#PhDStudent'/>
</rdf:Property>
```

*Example 6 describes the hierarchy of a subset of academic staff :*

- Lecturer, Researcher, PhDDirector and PhDStudent are *classes*.
- PhDDirector is a subclass of Lecturer
- PhDStudent is a subclass of Researcher

*As well as the property Supervisor, this applies to PhDDirector (domain) and applies to a PhDStudent (range).*

### 2.4.3 Web Ontology Language (OWL)

Through the RDFS standard, machines became increasingly able to collecting, interpreting and integrating data from different sources. Despite this, they remain still quite limited in what is possible to express with this vocabulary. For example, we can face a situation where we have to show that two IRIs reflect the same resource; or that two classes are complementary, similar or disjoint; or even that a property is symmetric, transitive or not; or to define a new class as the union/intersection of two existing classes; and many other similar potential situations. By Extending the RDFS vocabulary with a wide range of new well-defined terms, the Web Ontology Language[26] (OWL), another W3C standard, aims to respond to these (and many other) situations. Consequently, much richer semantics can be made explicit through this vocabulary enrichment, which makes

[26] https://www.w3.org/TR/owl-features/

possible a better automatic data integration from a variety of sources. It should also be noted that its second version, OWL2[27] allows to, compared to its predecessors, define more complex associations of resources as well as the properties of their respective classes.

OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full, so that any valid OWL Lite ontology is also a valid OWL DL ontology, and any valid OWL DL ontology is also a valid OWL Full ontology:

- ***OWL Lite*** is the simplest sub-language of OWL. While it supports cardinality constraints, it only allows cardinality values of 0 or 1. It is intended to express a simple concept hierarchy. Where can we use OWL Lite? For instance, in a situation where we are invited to perform rapid migrations from old thesauri, OWL Lite is the most suitable.

- ***OWL DL*** is more complex compared to OWL Lite. It allows a much greater expressiveness. As its name suggests, OWL DL is based on Description Logics and dedicated to supporting automated reasoning. It is designed to provide the maximum possible of expressiveness while taking into account the completeness of calculations and their decidability. Completeness of reasoning means that all inferences are computable, while decidability means that their computation is done in a finite time.

- ***OWL Full*** is the most complex version of OWL, and the one that allows the highest level of expressiveness. Based on a different semantics from OWL Lite or OWL DL, OWL Full was designed to preserve some compatibility with RDF Schema. For example, a class can be treated either as a set of individuals or as an individual in its own right in OWL Full, thing that is not allowed in OWL DL. In practice, OWL Full is used in situations where it is more important to have a high level of description capability, even if it means not being able to guarantee the completeness and the decidability of the calculations, which also means no reasoning software is able to perform complete reasoning for it. One of the most interesting mechanisms that OWL Full offers is the possibility of extending the default vocabulary of OWL.

[27] https://www.w3.org/TR/owl2-syntax/

There is an overabundance of explanatory works of the OWL and OWL2 vocabularies in the literature (Keet, 2018; Hogan et al., 2020; Tudorache, 2020… etc.). In this dissertation and in the same way as RDF and RDFS above, we do not detail all that the W3C has standardized as syntax for OWL and OWL2. By cons, to get an idea of the enrichment brought by OWL compared to RDF and RDFS, we unfold the following series of examples (Examples 7, 8, 9, 10, 11, 12 and 13). These examples are an illustration of OWL syntax applied to the creation of an ontology from the educational domain.

#### 2.4.3.1 Ontology structure

OWL originally relies on XML syntax, which makes it necessary to define a number of namespaces in the headers of OWL files. These namespaces will allow the use of identifiers unique for OWL and make ontologies easy to read by humans. In Example 7 we declare the URI of the ontology under construction.

***Example 7.*** *A namespace for an academic ontology:*

```
xmlns:university=http://www.example.com/ontologies/2021/university-
20210101/university#
```

Since ontologies are intended to be shared on the Web, the URI of an ontology is not only used to identify it but also it will allow this ontology to be called in another document. Therefore, it is possible to specify at the head of the OWL document, the version of the ontology, URIs of previous versions, comments,…, for several potential purposes, in particular to facilitate the reuse of created ontologies, extend them, versioning,…. Example 8 shows an URI of an old ontology version.

***Example 8.*** *A prior version URI of an academic ontology:*

```
<owl:priorVersion
rdf:resource="http://www.example.com/ontologies/2020/university20200101/universi
ty"/>
```

#### 2.4.3.2 Basic elements

Now, we define the ontology in itself. First, let's build classes from the domain to be described using this ontology, then properties on the individuals of these classes.

**Simple classes and individuals.** To create the root classes that interest us, we need to declare their names. Note that in OWL, every class is derived from the class "owl:Thing",

and there is also an empty class "owl:Nothing". Example 9 shows the creation of two classes.

***Example 9.*** *Declaration of two classes: Lecturer and Researcher.*

```
<owl:Class rdf:ID="Lecturer"/>
<owl:Class rdf:ID="Researcher"/>
```

We can then inherit other classes using the "rdf:subClassOf" identifier. Example 10 shows this.

***Example 10.*** *Declaration of PhDStudent class as a subclass of Researcher.*

```
<owl:Class rdf:ID="PhDStudent">
        <rdfs:subClassOf rdf:resource="#Researcher" />
</owl:Class>
```

Once we are done with ontology classes, we could define their individuals. It is enough to declare the individual name and its type. The type states of which class the individual is an instance. Example 11 shows the creation of an individual.

***Example 11.*** *Declaration of Younes as an instance of the PhDStudent class.*

```
<owl:Thing rdf:ID="Younes" />
<owl:Thing rdf:about="#Younes">
        <rdf:type rdf:resource="#PhDStudent"/>
</owl:Thing>
```

**Simple properties.** Once we have defined the classes and their contents, we need to define the properties on the individuals of these classes. These properties are what will allow the machine to reason about individuals. A property is defined by giving its domain (typically a class) and its range (which can be another class or an XML data type like xml:Integer). This gives us two categories of properties that an ontology builder may want to define:

- Object properties (owl:ObjectProperty) which link individuals to individuals.
- Datatype properties (owl:DatatypeProperty) which link individuals to data values.

Example 12 shows the creation of an object property.

***Example 12.*** *Declaration of the property SupervisedBy with the class PhDStudent as domain and the class Lecturer as range.*

```
<owl:ObjectProperty rdf:ID="SupervisedBy">
    <rdfs:domain rdf:resource="#PhDStudent"/>
    <rdfs:range rdf:resource="#Lecturer"/>
</owl:ObjectProperty>
```

Similarly, we can derive properties as we derive classes.

#### 2.4.3.3 OWL2 Axioms

Axioms are considered as the main component within an OWL2 ontology. An axiom is defined as a statement that says what is true in the domain. OWL2 provides an extensive set of axioms divided into three categories: Class expression axioms, Object property axioms and Data property axioms.

- **Class Expression Axioms** allow relationships to be established between class expressions. This set consists of four different axioms: {SubClassOf, EquivalentClasses, DisjointClasses, DisjointUnion}.
- **Object Property Axioms** can be used to characterize and establish relationships between object property expressions. This set consists of thirteen different axioms: {SubObjectPropertyOf, EquivalentObjectProperties, DisjointObjectProperties, InverseObjectProperties, ObjectPropertyDomain/Range, FunctionalObjectProperty, InverseFunctionalObjectProperty, Reflexive/IrreflexiveObjectProperty, Symmetric/AsymmetricObjectProperty, TransitiveObjectProperty}.
- **Data Property Axioms** can be used to characterize and establish relationships between object property expressions. This set consists of six different axioms: {SubDataPropertyOf, EquivalentDataProperties, DisjointDataProperties, DataPropertyDomain/Range, FunctionalDataProperty}.

Although we only list them, we do not detail here the meaning of each of the OWL2 axioms. However, for more details the reader is invited to consult the Axioms section[28] in the OWL2 Structural Specification and Functional-Style Syntax. Besides, Example 13 is given to clarify the principle.

[28] https://www.w3.org/TR/owl2-syntax/#Axioms

***Example 13.*** *Consider an ontology consisting of the following axioms.*

| | |
|---|---|
| SubObjectPropertyOf( a:hasPhDPosition a:hasJob ) | Having a PhD Position implies having a Job. |
| ObjectPropertyAssertion(a:hasPhDPosition a:Younes ) | Younes has a PhD Position. |

*Since a:hasPhDPosition is a subproperty of a:hasJob, each couple of individuals connected by the former property expression is also connected by the latter property expression. Therefore, this ontology entails that a:Younes is connected to the property a:hasJob; that is, the ontology entails the following assertion:*

ObjectPropertyAssertion(a:hasJob a:Younes )

Comparing to Example 5, we exclude the three mentioned conditions to infer that "Younes has a job", namely: First, the reader (user) does not have to be human since the content is now in a machine-readable form; secondly, the user would not be frozen to understand only English because there is precise semantics of the data independent of human languages; and finally, there is no restriction (at least for the size) to extend an ontology according to the formalization needs.

## 2.5 Ontology Validation

### 2.5.1 OWL property restrictions

The novelty that OWL has brought over its predecessor RDFS is the notion of the restriction on properties. This helps build a powerful inference language, since OWL restrictions are not really data constraints, but rather describe inferences to be applied based on them. If we take for example, the restriction owl:maxCardinality 1 stating that a person can only have 1 value for a:hasFather and a:Younes an instance of a:Person that has two a:hasFather values, then an OWL reasoner will assume that these two values must in fact represent the same real-world entity, just with different URIs. Such situations are confirmed when there is a need to check whether a set of instances conforms to a given schema. The OWL reasoner would not be able to answer true or false in the same way as an XML Schema validator about an XML file. Instead, this reasoner will actually add to the data in attempt to conform to the restrictions rather than report an error.

### 2.5.2 Techniques before SHACL

As a solution to this kind of problem, tools supporting OWL (such as Protégé[29] and TopBraid[30]) provide data entry forms that restrict users from entering data that do not comply with the stated restrictions. For the same purpose, SPARQL[31] queries are also considered as a solution for testing RDF graphs with the Where clause. Although SPARQL does not make any assumptions about inferences existence or not, this kind of query simply interrogates the triples that are actually asserted in the data. This way of using SPARQL as a language to constraint data allowed the creation of an RDF vocabulary called SPARQL Inferencing Notation[32] (SPIN), also known as "SPARQL Rules". SPIN defines the properties that can be used to attach SPARQL queries to classes, indicating that all instances of these classes must meet the constraints stated by those SPARQL queries. Although SPIN is not an official W3C standard which minimizes its chances of achieving widespread industry adoption, it has become popular among a large user community.

### 2.5.3 Shapes Constraint Language (SHACL)

RDF Data Shapes Working Group[33] was launched by W3C in 2014 with the aim of making up for the lack of a suitable standard to express constraints and schemas. Based on SPIN and other member submissions[34], this group was able to standardize the Shapes Constraint Language[35] (SHACL) as a W3C Recommendation in July 2017. SHACL is a language for validating RDF graphs against a set of conditions. There are two types of graphs in SHACL: (*i*) *Shapes Graphs* which express a set of shapes (a collection of constraints that apply to targeted RDF resources) and other constructions, allowing to provide the set of target conditions, and (*i*) *Data Graphs* which represent the RDF graphs that are validated against a shapes graph, and the operation produces a validation report, also expressed as a graph. All these graphs can be represented in any RDF serialization formats including JSON-LD or Turtle (see Section 2.4.1).

---

[29] http://protege.stanford.edu/
[30] http://www.topquadrant.com/topbraid/
[31] http://www.w3.org/standards/techs/sparql
[32] https://spinrdf.org/
[33] https://www.w3.org/2014/data-shapes/wiki/Main_Page
[34] https://www.w3.org/Submission/shapes/
[35] https://www.w3.org/TR/shacl/

Although the first objective of the SHACL shapes graphs was to validate that data graphs satisfy a set of conditions, they are also used as a description of the data graphs that do satisfy these conditions. In addition to the validation, SHACL descriptions can be used for various purposes, such as data integration, code generation and user interface building.

The SHACL users not only benefit from the ability to specify a severity level for the validation results, but also the ability to consider suggestions on how the data can be corrected depending on the validation result. Three levels of severity are proposed: *Violation* (by default if no sh:severity has been specified for a shape), *Warning* and *Info*. In addition, SHACL users can add other, custom levels of severity.

## 2.6 Ontology Evolution & Versioning

In this dissertation, we are not really interested in the ontological change[36] process, but rather in the potential results of this process. Such a process normally produces a set of ontological axioms resulting from the changes applied to a version of an ontology. The axioms representing ontological change will be useful for developing our proposition later. However, it is crucial in this section to define the ontology evolution and versioning concepts and to differentiate between them.

### 2.6.1 Ontology evolution

The dynamism in science and the business conditions evolution force ontologies to change in order to support new contexts and requirements. In addition, several problems arise when trying to use independently developed ontologies together, or when existing ontologies have to be modified to accommodate new goals. A such domain of research is known as Ontology Management. It requires a set of methods and techniques allowing the efficient use of multiple ontologies formed from different sources, modifying ontologies according to new requirements or maintaining different variants of the ontology, etc. Indeed, ontology evolution is considered as a discipline of ontology management facilitating the modification of an ontology while preserving its consistency. Based on the existing research on the evolution of database schemas, Stojanovic (2004) defines ontology

[36] For more details, the reader is prompted to return to section 3.3 in (Zahaf, 2017).

evolution as: "*the timely adaptation of an ontology to the arisen changes and the consistent propagation of these changes to dependent artifacts*".

### 2.6.2 Ontology versioning

Some current works are based on research on schema evolution and versioning in object-oriented and relational databases to consider ontology version management as a variant of ontology evolution (Haase & Sure, 2004). Under this point of view, ontology evolution is concerned with the ability to modify an ontology without loss of data and allowing access to data via the latest ontology version, while the ontology version management allows to create different versions and to access the data through these versions. From another point of view, Flouris (2006) differentiates between version management and ontology evolution. He considers the evolution as the process of modifying the ontology while maintaining its validity, while version management is the process by which multiple versions of the same ontology are managed while maintaining interoperability between these versions and allowing access to each version according to the requirements of the access element (data, service, application or another ontology).

## 2.7 Ontology Alignment

### 2.7.1 Introduction

Ontology alignment is the task to detect links between elements from two ontologies. These links are referred as correspondences and express semantic relations between ontological entities. Two entities are matched when it is asserted that a correspondence between such two entities exists, with regard to the considered semantic relation. While a matchable element can be any arbitrary entity, in this dissertation we consider only alignments of matchable entities that belong to ontologies. We adapt the definition of Euzenat & Shvaiko (2007) as follows:

**Definition 2.3** (Ontology alignment). Given two ontologies $O_i$ and $O_j$, let $Q(O_i)$ (respectively $Q(O_j)$) be the set of matchable entities of $O_i$ (respectively $O_j$). A correspondence between $O_i$ and $O_j$ is a 4-tuple $(e_i, e_j, r, n)$ such that, $e_i \in Q(O_i)$, $e_j \in Q(O_j)$, $r$ is a semantic relation, and $n \in [0; 1]$ is a confidence value. An alignment $M$ between $O_i$ and $O_j$ is a set of correspondences between $O_i$ and $O_j$. We restrict $r$ to be one of the semantic relations from the set $\{\text{Equivalence}(\equiv), \text{Subsumption}(\sqsubseteq), \text{Disjunction}(\perp)\}$.

The confidence value used in mappings intuitively corresponds to the confidence that the mapping holds, where the confidence increases towards value 1. Figure 2.9 shows the correspondences between two educational domain ontologies as well as the related confidence values.

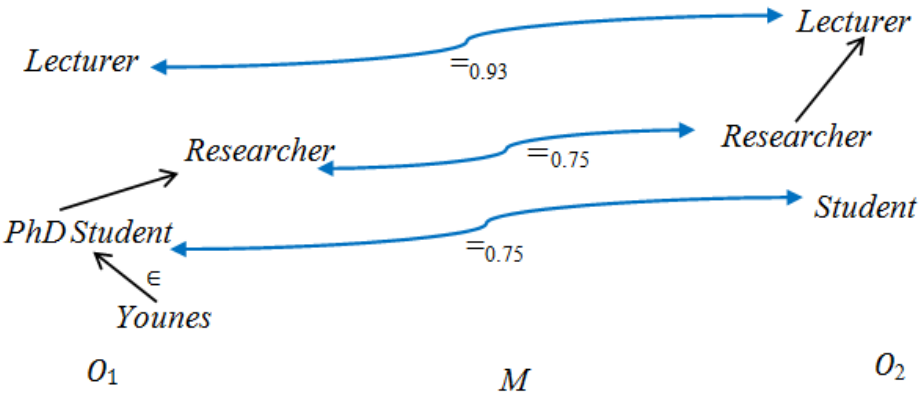


**Figure 2.9. An alignment M between two educational domain ontologies $O_1$ and $O_2$**

***Example 14.*** *Considering the alignment $M$ of Figure 2.9. We use DL like syntax to describe both ontologies. Also, we use the index number in ontologies notation as name space to designate entities. Alignment M is created by the ontology matching system YAM++*[37].

$$O_1 = \begin{Bmatrix} PhDStudent \sqsubseteq Researcher \\ PhDStudent(\text{Younes}) \end{Bmatrix};\ O_2 = \{Researcher \sqsubseteq Lecturer\}$$

$$M = \begin{Bmatrix} 1: PhDStudent =_{0.75} 2: Student \\ 1: Researcher =_{0.75} 2: Researcher \\ 1: Lecturer =_{0.93} 2: Lecturer \end{Bmatrix}$$

There is no standard for alignment semantics. In (Borgida & Serafini, 2003), distributed description logics semantics have been proposed. Another approach, called reductionist semantics, interprets correspondences of the alignment as axioms in some merged ontology (Meilicke & Stuckenschmidt, 2009). The merged ontology is called aligned ontology. In this dissertation, we use an example of this semantic called natural semantic. It involves building a merged ontology through the union of the two ontologies to align and axioms obtained by translating relations of the alignment. We introduce this semantic through its aligned ontology.

[37] http://www.lirmm.fr/yam-plus-plus/

**Definition 2.5** (Natural Semantics). Given an alignment $M$ between two ontologies $O_1$ and $O_2$ and trans: $M \longrightarrow A$, a function that transforms a correspondence to an axiom. The natural semantics of $M$ is defined by the following aligned ontology:

$$O_1 \cup_M O_2 = O_1 \cup O_2 \cup trans(M).$$

***Example 15.*** *Following example 14, the transformation of the alignment M to axioms is as follows:*

$$trans(M) = \left\{ \begin{array}{c} 1: PhDStudent \equiv 2: Student \\ 1: Researcher \equiv 2: Researcher \\ 1: Lecturer \equiv 2: Lecturer \end{array} \right\}$$

We introduce the notion of alignment consequence according to natural semantics as follows:

**Definition 2.6** (Alignment consequence). An axiom $\delta$ is an alignment consequence of an alignment $M$ between two ontologies $O_1$ and $O_2$ if and only if $\delta$ is a logical consequence of the aligned ontology $O_1 \cup_M O_2$. We denote this relation by $O_1 \cup_M O_2 \models \delta$.

An axiom which is an alignment consequence either represents an ontological axiom or the image of a correspondence by the transformation function of the alignment.

***Example 16.*** *Following example 15, it is clear that $O_1 \not\models PhDStudent \sqsubseteq Lecturer$ but since, $O_1 \cup_M O_2 \models$ 1:Researcher $\equiv$ 2:Researcher, 1:PhDStudent $\sqsubseteq$ Researcher, 1:Lecturer $\equiv$ 2:Lecturer, we can derive that $O_1 \cup_M O_2 \models$ 1: PhDStudent $\sqsubseteq$ Lecturer.*

**Definition 2.7** (ontology signature isomorphism). Given two ontologies $O_1=(S_1, A_1)$ and $O_2=(S_2, A_2)$, an ontology signature isomorphism is a particular alignment $M: S_1 \rightarrow S_2$ such that $A_2 \models M(A_1)$ and $A_1 \models M^{-}(A_2)$, i.e., all models of $A_2$ are models of the image of $A_1$ by $M$ and vise versa. The image of an axiom is obtained by systematically replacing signature elements of this axiom by their correspondents, according to the signature isomorphism $M$.

***Example 17.*** Following example 15, the set of signature isomorphism of the ontology $O_2$ by the alignment $M$ is: $\left\{ \begin{array}{c} 2: Student \models M(1: PhDStudent) \\ 2: Researcher \models M(1: Researcher) \\ 2: Lecturer \models M(1: Lecturer) \end{array} \right\}$

### 2.7.2 Ontology alignment life cycle

According to (Euzenat et al., 2008), three consecutive phases constitute the life cycle of the alignment between ontologies, namely: The design phase, the sharing phase and the operating phase. Figure 2.10 is an adaptation of the works in (Euzenat & Shvaiko, 2013) to illustrate these phases and the related tasks. The conception phase is an iterative process formed by three tasks: the creation task, the evaluation task, and the enhancement task.

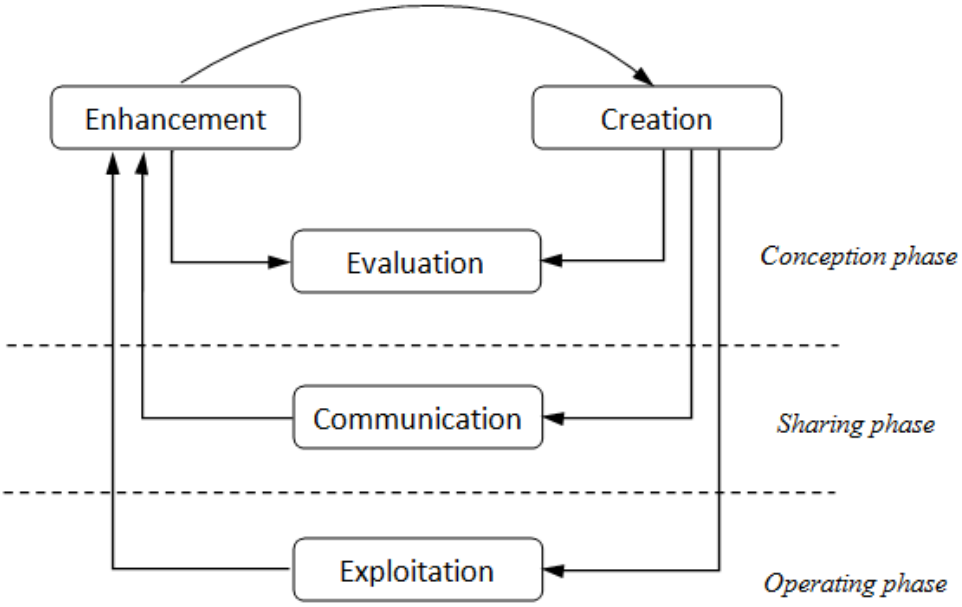


**Figure 2.10. The ontology alignment life cycle**

The first task of the alignment life cycle is the creation task. It is also known by the ontology matching task and aims to create alignments by calculating semantic links between two ontologies. Several efficient ontology matching tools have been available in the recent years (Euzenat & Shvaiko, 2013). They differ basically in the nature of the knowledge encoded in the ontology and the methods used in the identification of correspondences (Euzenat et al., 2011). Terminological methods compare the lexicon used to designate ontological entities, while structural methods consider the internal and/or external structure of the ontology. Some approaches are based on model theory to compute correspondences between input ontologies entities. The ontology extension can also be used. Almost all existing matching systems combine these techniques to fulfill lacks of every category type. The obtained alignment from these tools may be subject of bugs and to be useful, it should be evaluated. The second task of the alignment life cycle is the evaluation task. It consists of assessing the correctness as well as the completeness of the alignment which might lead to an enhancement. The third task is the enhancement task

which may be the subject of a debugging process if the alignment contains erroneous correspondences, an adapting process following an ontology change, enhancing an incomplete alignment, or just a call of refinement procedures such as the alignment trimming relatively to a fixed threshold. These three tasks might then go through an iterative process until an alignment is deemed worth publishing.

The sharing phase includes the activities of storing and communicating the alignment to other parties interested in such an alignment. Nowadays, a set of open servers are available, such as for instance Bioportal (http://bioportal.bioontology.org), AgroPortal (http://agroportal.lirmm.fr/), and Alignment server (http://alignapi.gforge.inria.fr/aserv.html), which manage alignments as first class citizens in order to store, index, organize and share them. Users can browse alignments, upload new alignments, and download alignments that the repository has (Noy et al., 2008).

Finally, the operating phase allows to exploit the alignment. Also during this phase, alignment servers can deliver it in different formats which allows to expand its usefulness. Consumer applications subsequently interpret and use the alignment according to their needs and actions, like mediation and merging.

### 2.7.3 Alignment evolution

Alignment Adaptation, Alignment Maintenance, Alignment Evolution and Alignment Revision, all reflect the names used in the literature to refer to the alignment evolution problem (Dos Reis et al., 2015). This discipline attempts to correct ontology alignment during the third task of the first phase of the alignment life cycle (see Section 2.6.2).

Alignment revision was the objective of Euzenat (2015). The author considers ontologies as logical theories to study the problem of restoring consistency of a network of ontologies formed by a set of ontologies connected by a set of alignments when concerned ontologies were evolved or the alignment was improved by adding some correspondences.

Software development domain commonly uses debugging to refer to the operation performed before the delivery of the final product. By considering ontologies and alignments as software products, alignment debugging is defined as a task performed before alignment delivery to diagnose and repair alignment produced by ontology

matching tools, since created alignments might contain errors such as redundancy, inconsistency, imprecision, or an abnormal behavior (Wang & Xu, 2008).

## 2.8 Conclusion

The intention of this chapter was to prepare the reader for the context studied in this dissertation. At the beginning, we expressed the different points of view in the definition of ontology, which resulted in a spectrum of semantic expressiveness levels for the different ontological forms (Uschold & Gruninger, 2004). Afterward, we have exposed a set of ontology-based applications emerged in recent years. These applications, and as research continues to develop, have experienced a growing need to involve semantic interoperability solutions through the notion of ontology alignment.

In a hierarchical manner, we have presented, in a concise but relevant way, the technologies of the Semantic Web. RDF, RDFS and OWL have all emerged as W3C standards to allow better formalization of knowledge circulated on the classic Web. This consequently makes it possible to launch a new generation of the Web, which takes advantage of the ascending computing power of recent machines by putting data in machine-readable forms.

Although, we are not interested here in the ontological change in itself, we have briefly discussed two important concepts in this context, namely: ontology evolution and ontology versioning. These two notions are upstream of the problematic studied in this dissertation, since their outputs serve as a kind of trigger to deal with the alignment repair problem following the evolution of one of its input ontologies (see Chapter 4) .

At the end, we have clarified the ontology alignment notion, its syntax and semantics as well as its life cycle to specify later at which level of this cycle we intervene to apply our proposal. We also shed light on concepts used in the literature to refer to the task of evolving ontology alignment so that it meets new requirements or demands.

# Chapter 3: Problem Statement & Related Works

## 3.1 Introduction

The present chapter aims to make the link between what has already been achieved as solutions to the alignment adaptation problem, and what we aim to solve under this issue. This allows us at the end, to position ourselves in relation to the existing techniques. For this purpose, we formalize the problematic dealt with in this dissertation in Section 3.2. With a concrete example, we try to illuminate the issues of the *Alignment Conservativity Under Ontology Change*. We then review the related literature in Section 3.3. This part has two subsections to explore the state of the art under two contexts: *Ontology Matching* context and *Alignment Adaptation* context. Finally, section 3.4 concludes this examination and allows us to position our work against existing methods.

## 3.2 Problem Statement

The work of Zahaf & Malki (2016) is based on the belief base revision theory to introduce two postulates, namely: ontology change preservation and logical consistency for alignments repair following the ontology change. The change preservation ensures that deleted axioms should no longer be logical consequences of the alignment. While logical consistency guarantees that ontological change does not generate contradictory knowledge in ontologies. Note that in monotonic logics, an inconsistency can only occur if certain types of axioms have been added. We reconsider these two postulates in the context of the conservativity principle under ontological change. We reformulate the former and generalize the latter to integrate any type of added axioms, and we define the general concept of conservation in the context of alignment evolution under ontology change. In this context, the alignment is conservative if the ontological change does not have to introduce new semantic relationships between the concepts of an introductory ontology.

Jiménez-Ruiz et al. (2011) identified the conservativity principle as a conservativity extension (Lutz et al., 2007) problem by calculating the deductive difference between one

ontology and its extension consisting in adding alignment to this ontology (i.e., $diff(O_i, O_i \cup M)$). Solimando et al. (2016) generalize this proposition and state that deductive difference $diff(O_i, O_i \cup_M O_j)$ between any ontology $O_i$, such that $i \in \{1, 2\}$, and the aligned ontology must be empty with regard to the signature of that ontology. The deductive difference between $O_i$ and $O_i \cup_M O_j$ is the set of entailments $\{\delta\}$ formulated over $Sig(O_i \cup O_j)$ that do not hold in $O_i$, but do hold in $O_i \cup_M O_j$. Formally:

$$diff(O_i, O_i \cup_M O_j) = \{\delta \mid O_i \nvDash \delta \text{ and } O_i \cup_M O_j \vDash \delta \text{ and } Sig(\delta) \subseteq Sig(O_i \cup O_j)\}$$

We differentiate two possible situations in which the alignment can fall into conservativity violation depending on whether the violation appeared before or after the ontological change. In the context of the evolution of alignment under the ontology change, we are concerned with the second situation, that is, the violation of conservativeness caused by the ontological change. Thus, we define the alignment conservativity violations under ontology change as the set theoretical difference between the alignment conservativity violations before and after the change. Formally,

**Definition 3.1** (Alignment Conservativity Under Ontology Change). Let $O_{i1}$ *and* $O_{i2}$ two versions of the evolved ontology $O_i$. $M$ an alignment between two ontologies $O_i$ and $O_j$ is conservative under ontology change if and only if there are no violations after the change, except for those before the change:

- $diff(O_j, O_{i2} \cup_M O_j) = diff(O_j, O_{i1} \cup_M O_j)$
- $diff(O_{i2}, O_{i2} \cup_M O_j) = diff(O_{i1}, O_{i2} \cup_M O_j)$

Example 18 shows two conservativity violation situations: Figure 3.1 illustrates the problem before the change and Figure 3.2 illustrates it after the change.

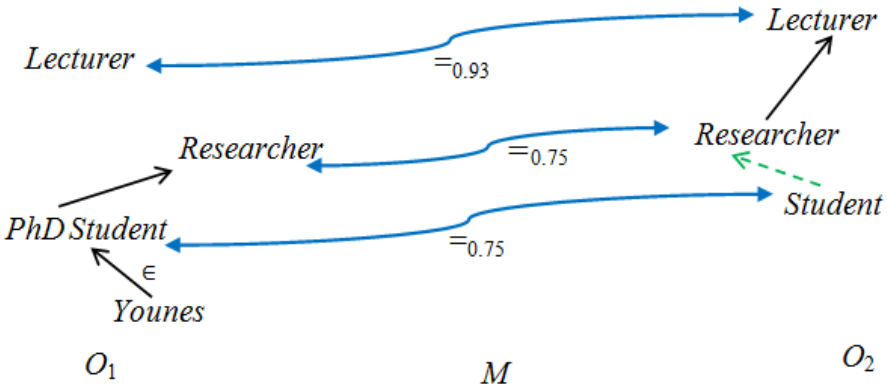


**Figure 3.1. Conservativity violation before ontology change**

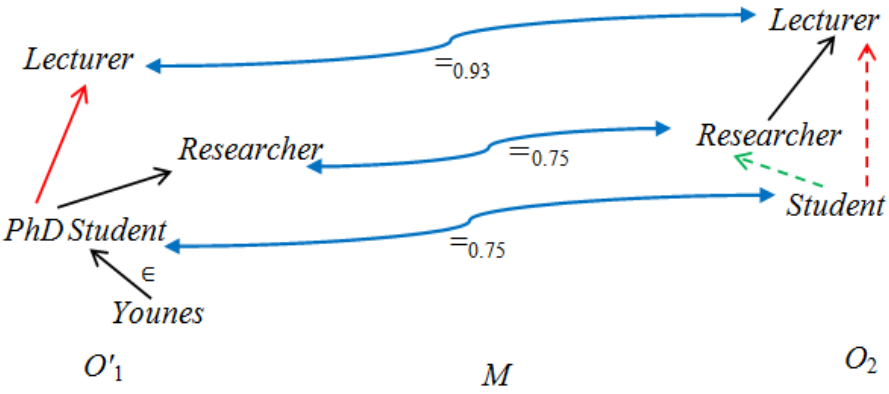


**Figure 3.2. Conservativity violation after evolving the ontology $O_1$ to $O_1'$**

***Example 18.*** *Following Example 14, we note that, before the evolution of the input ontologies as illustrated in Figure 3.1, the set of conservativity violations is the deductive difference diff($O_2$, $O_1 \cup_M O_2$) = {2:Student ⊆ 2:Researcher}. Therefore, the axiom 2:Student ⊆ 2:Researcher (dashed green arrow) represents a violation of the conservativity before the change..*

*Assuming now that one of the two input ontologies has been evolved and let $O_1'$ = $O_1 \cup$ {1:PhDStudent ⊆ 1:Lecturer} be the new version of $O_1$ following the addition of the new axiom 1:PhDStudent ⊆ 1:Lecturer (solid red arrow). This change can be requested for example by applications using ontology $O_1$, since the added axiom is entailed when using $O_1$ in conjunction with $O_2$ and alignment M, which leads ontology $O_1$ owners to explicitly evolve it by adding a new axiom {1:PhDStudent ⊆ 1:Lecturer}. In this situation, diff($O_2$, $O'_1 \cup_M O_2$) ={{2:Student ⊆ 2:Lecturer}, {2:Student ⊆ 2:Researcher}} ≠ diff($O_2$, $O_1 \cup_M O_2$). So, according to definition 3.1, alignment M violates the conservativity under evolving $O_1$.*

## 3.3 Related Works

Recently, many approaches have appeared to solve the problem of alignment evolution under the change of ontologies. We can identify two types of classes: approaches that

calculate the new alignment from scratch by using ontology matching tools, and approaches that reuse as much as possible the old alignment by adapting it according to the ontology change. The main challenge for approaches of both types is to maintain the consistency of alignment after applying the change (Euzenat, 2015). An alignment is consistent if and only if the ontologies remain consistent even when used in conjunction with the alignment. Haase & Stojanovic (2005) distinguish three types of consistency: structural, logical and user defined consistency. The structural consistency is determined by a set of conditions with respect to the underlying models of ontologies, while logical consistency ensures no contradiction can be entailed from those ontologies. The user-defined consistency refers to user requirements that need to be expressed outside of the ontology language itself. Other methods like (Jiménez-Ruiz et al., 2011), (Solimando et al., 2014a), (Solimando et al., 2014b) and (Solimando et al., 2016) have taken a step forward to treat the conservativity of alignment. The conservativity is a general form of logical consistency which prevents any unwanted axioms from being a logical consequence of the alignment. To our knowledge, no method has previously addressed the conservativity problem under ontological change. In this context, an alignment is conservative if the ontological change should not introduce new semantic relationships between concepts from one of the input ontologies.

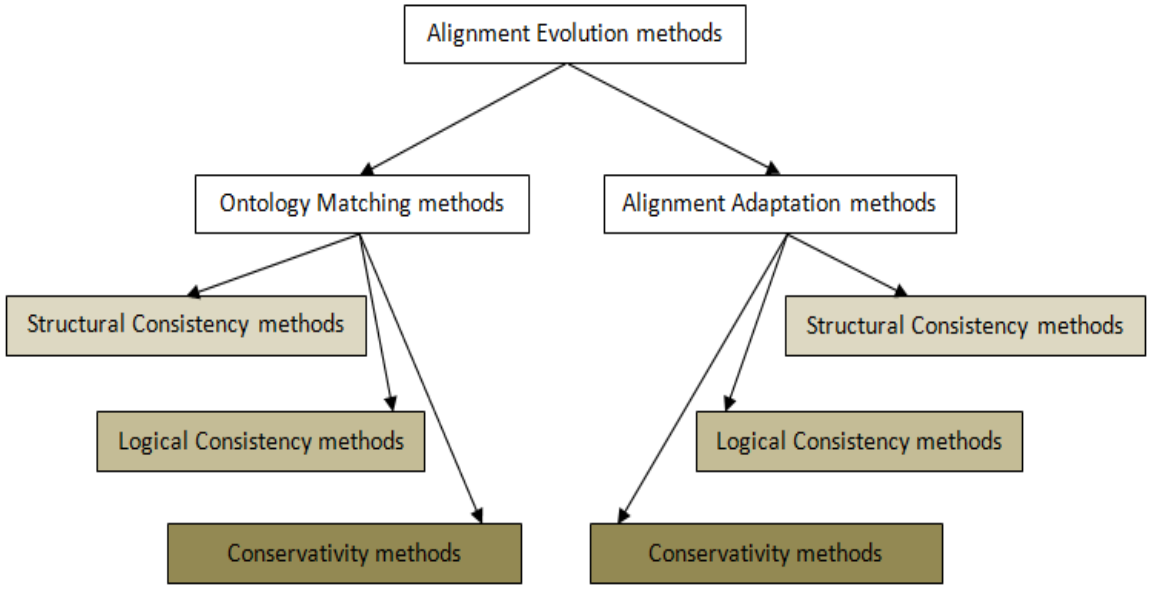


**Figure 3.3 Classification tree of alignment evolution under ontology change methods**

In the following, we discuss the approaches of the two classes according to the type of consistency they ensure during the evolution of the alignment. We first explore in Section

3.3.1, the left branch according to Figure 3.3, to examine alignment evolution methods under ontology change in the ontology matching context, while section 3.3.2 analyzes alignment evolution methods under ontology change in the ontology adaptation context.

### 3.3.1 Ontology matching methods

We consider ontology matching methods as a solution to deal with alignment evolution under ontology change by calculating a new alignment from scratch. Basically, ontology matching tools differ in the nature of the knowledge encoded in the ontology, and the techniques used in the identification of correspondences (Euzenat et al., 2011). Terminological techniques compare the lexicon used to designate ontological entities, while structural techniques consider the internal and/or the external structure of the ontology. Some approaches are based on the model theory to compute correspondences between input ontologies entities. Ontology instances can also be used. Almost all existing matching systems combine these techniques to fulfill lacks of every category type and maintain the alignment consistency after applying the change (Euzenat, 2015). An alignment is consistent if and only if the ontologies remain consistent even when used in conjunction with the alignment. Note that in the literature, the notion of consistency is remedied according to two levels: structural consistency and logical consistency. Structural consistency ensures that alignment obeys the constraints of its underlying representation structure (Martins & Silva, 2009), while logical consistency considers the semantics of the alignment. An alignment is logically consistent if and only if it preserves the satisfiability of ontologies (Zahaf & Malki, 2016), meaning that it does not introduce contradicting knowledge in ontologies.

#### 3.3.1.1 Systems dealing with structural consistency

Structural consistency is targeted by a first set of tools like ALIN (Da Silva et al., 2020) through an interactive phase based on expert feedback to produce the so called mappings suggestions. After each expert feedback, ALIN modifies the set of mapping suggestions using the structural analysis of ontologies and alignment anti-patterns. Despite the excellent consistency and the conservativity results marked in the OAEI 2019[38] campaign, ALIN remains semi-automatic and very dependent on expert feedback correctness which

[38] http://oaei.ontologymatching.org/2019/results/conference/index.html#logical

makes it unsuitable in the fly. SANOM (Mohammadi et al., 2019) combines lexical (Jaro-Winkler and WordNet) and structural metrics to map entities of two ontologies. It seems effective in dealing with structural consistency, but has no guarantees towards logical consistency.

#### 3.3.1.2 Systems dealing with logical consistency

For the same purpose of calculating a new alignment following the ontological change, a second set of approaches tries to guarantee a logical consistency in their results. For example, Lily's (Wang & Xu, 2008) authors define two types of inconsistencies: (*i*) Mappings that form a circle and (*ii*) Mappings that do not meet the *equivalentClass*/*disjointWith* axioms mentioned in the input ontologies $O_1$ and $O_2$. They use an algorithm that combines these ontologies (the alignment between them is a single graph (*is-a*)), and detects the paths which constitute a circle to inform the user of inconsistent mappings. Regarding the reparation, Lily treats all suspicious mappings like program debugging in two categories: errors and warnings. Apparently, errors are the confirmed wrong mappings, but warnings are the ones which may be wrong, right or imprecise. There are two proposed solutions for the two previously mentioned types of detected inconsistencies. In the first type, paths which constitute a circle are considered as wrong. The choice to delete one of the arcs forming the circle is left to the user. In the second type, Lily proposes two potential solutions: (*a*) Importing a complex concept and representing the mappings in the form: $m$: $e_1 \equiv e_2 \vee e'_2$, such as: $(e_1) \in O_1$ and $(e_2, e'_2) \in O_2$. (*b*) Giving the user the choice to delete one of the mappings in conflict. Note that Lily considers only the mappings between concepts and only *equivalentClass*/*disjointWith* as axioms.

YAM ++ (Ngo & Bellahsene, 2012) is based on the ALCOMO[39] system to debug alignment. Meilicke, the author of ALCOMO (Meilicke & Stuckenschmidt, 2009), readjusts the notion of Minimal Incoherency Preserving Sub-TBox (MIPS) identified in ontology debugging (Schlobach & Cornet, 2003) to the notion of MIPS (Minimal Incoherence Preserving Sub-alignment) and MUPS (Minimal Unsatisfiability Preserving Sub-alignment), to detect inconsistency and unsatisfiability in alignment. He proposes a variant algorithm (expand-and-shrink-algorithm) with two reasoning components

[39] http://web.informatik.uni-mannheim.de/alcomo/

(complete and incomplete) for debugging incoherent alignments. The first is based on a pattern to detect all MIPS of an alignment *A*. However, this approach detects a large amount of conflicting pairs of correspondences between two input ontologies $O_1$ and $O_2$. The basic idea is to first classify these ontologies using an OWL2 reasoner. Given two alignment axioms $e_1 \equiv e_2 \in A$ and $e'_1 \equiv e'_2 \in A$ with $(e_1, e'_1) \in O_1$ and $(e_2, e'_2) \in O_2$, Alcomo checks if $O_1 \models e_1 \sqsubseteq e'_1$ and $O_2 \models e_2 \sqsubseteq \neg e'_2$. If so, then $O_1 \cup O_2 \cup A \models e_2 \sqsubseteq \bot$, i.e., $e_2$ is unsatisfiable in the aligned ontology via *A*. Therefore, the set of correspondences $\{e_1 \equiv e_2, e'_1 \equiv e'_2\}$ is inconsistent. This idea is extended and four patterns are defined to take into account the subsumption and equivalence correspondences between classes and properties. These techniques can be accompanied by complete reasoning techniques. The related suggestion of such a combined approach is to compute a preliminary superset of a solution based on incomplete reasoning techniques. This intermediate result is then verified with complete reasoning techniques and further reduced if necessary. If full reasoning techniques are activated, it can be guaranteed that Alcomo generates a coherent set of correspondences by removing a repair *R* (a diagnosis) from *A*. Without activating complete reasoning techniques, Alcomo calculates an approximate repair $R^{\approx}$ (a subset of the diagnosis) and cannot guarantee the consistency of all output correspondences. The quality of a diagnosis can be defined in terms of aggregation of its confidence values. An intuitive idea is to remove the sets of correspondences with less aggregated values. In addition to the reasoning problem of detecting and repairing inconsistent correspondences, Alcomo aims to solve the problem of optimizing the proper consideration of the confidence values. For this, two different types of diagnosis have been defined: (*i*) A Global Optimal Diagnosis, which removes the slightest amount of confidence. If all correspondences are weighted equitably against their (positive) confidence values, an Optimal Global Diagnosis will be a diagnosis with the minimum number of correspondences. This type of diagnosis, however, is calculated by an exhaustive search algorithm, which will be impossible for large repair problems. (*ii*) The second type is called an Optimal Local Diagnosis, which can be constructed by a simple gluttonous approach starting with a set of empty correspondences *A′* which is extended step by step by adding all correspondences of *A*. Similar to our order relation in the diagnosis, these correspondences are decreasingly ordered according to their confidence values. Whenever a correspondence is added to *A′*, the consistency is checked through a combination of pattern-based techniques and complete reasoning. If *A′* becomes incoherent, the correspondence is not added.

ASMOV (Jean-Mary et al., 2009) introduces the notion of mapping validation, a graph built from the alignment and ontologies information. Two different constructs constitute this graph: nodes and edges. The nodes contain pairs of entities, whereas the edges contain pairs of properties. The validation process is done in three phases: concept validation, property validation and concept-property validation. In the first two phases, the considered edges (i.e., three types: *is-a, same-as* and *disjoint-from*) are created using the predefined properties of the ontology. The validation of the graph is reduced to an investigation of edges violation; a node may not be valid if one or more of the edges are violated. If an edge violation exists, only the linked nodes are investigated. In the third phase, the concept validation graph is modified. All edges are dropped from the remaining valid nodes and are replaced by edges created from the valid nodes of the property validation graph. The new graph is then validated, but in this time the nodes are favored; thus, only the edges are invalidated. All invalid mappings that have been identified are added to the invalid mapping list. If at least one violation was identified, the iteration process resumes and the invalid source-target pairs are ignored.

Matching systems presented so far are more or less efficient concerning alignment logical consistency violations. They deal with this problem according to the notion of contradictory axioms. This notion causes unsatisfiability within a set of alignment correspondences. However, this performance does not ensure the conservativity of the alignment following the changes in the related ontologies. In the context of alignment evolution under ontology change, an alignment is conservative if the ontological change should not introduce new semantic relationships between concepts from one of the input ontologies. These relationships are considered as violations of conservativity principle following ontological changes. As a response to this situation, a set of approaches have emerged. The aim of these approaches is to calculate a new alignment from scratch with respect to conservativity principle. We consider these approaches as a third set of ontology matching methods to deal with the problem of alignment conservativity under ontology change.

#### 3.3.1.3 Systems dealing with conservativity principle

The authors in (Jiménez-Ruiz et al., 2011) use, in their tool ContentMap, a specific pattern to detect conservativity principle violations. The pattern is based on the following observation: the OWL2 alignment *M* that encodes the contents of a specific thesaurus

(UMLS-Meta[40]) contains only axioms of the form *EquivalentClasses*($e_1$ $e_2$), where $e_1 \in O_1$ and $e_2 \in$ in $O_2$. This observation is used to significantly simplify the problem in the following way: $O_1$ violates conservativity *iff* there exist axioms *EquivalentClasses*($e_1$ $e_2$) and *EquivalentClasses*($e'_1$ $e_2$) in $M$, with $e_1$ and $e'_1$ different entities in $O_1$, such that $O_1$ alone does not imply the axiom *EquivalentClasses*($e_1$ $e'_1$). In this case, the mappings *EquivalentClasses*($e_1$ $e_2$) and *EquivalentClasses*($e'_1$ $e_2$) from $M$ are in conflict, and one of them may be incorrect. In order to identify such conflicting mappings, it suffices to syntactically check in $M$ whether two entities from one of the sources are mapped to the same entity in the other source, and then check semantically, using an ontology reasoner, whether these two entities were already equivalent with respect (only) to the former source. In order to disambiguate all the conflicts between two input ontologies, the authors use a diagnosis to remove the mapping with the smallest confidence value in each conflict. This technique is similar to ours in the choice of correspondences to be eliminated, but different in the size of conflict sets. While it is applied to pairs of mappings, our diagnosis deals with larger conflict sets. Since UMLS-Meta does not assign a confidence value to each mapping, the locality principle[41] is proposed to compute a confidence value for each conflicting correspondence. In the case where the locality principle doesn't hold, the authors identify three situations: (*i*) $M$ may be incomplete and new correspondences must be discovered. (*ii*) The definitions of the two concepts in their respective ontologies may be different or incompatible. (*iii*) The correspondence between $e_1$ and $e_2$ may be wrong. Although this approach can be seen as a solution to the alignment conservativity under ontology change problem, it suffers from two major drawbacks: firstly, the unique type of equivalence relation in the considered alignment which excludes the others possible types, and secondly, it takes only the ontology source and the alignment. Yet, this can be a subject of many neglected logical consequences when discarding the target ontology, since the lack of an asserted correspondence *EquivalentClasses*($e'_1$ $e_2$) in $M$ does not mean the lack of a derived relation between $e'_1$ and $e_2$. To argue this, we have provided a counterexample in a previous work (Atig et al., 2016).

[40] http://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html

41 If two entities e1 and e2 from ontologies O1 and O2 are correctly mapped, then the entities semantically related to e1 in O1 are likely to be mapped to those semantically related to e2 in O2 (Jiménez-Ruiz et al., 2011).

Another variant of the conservativity principle was cited in (Solimando et al., 2016), where the aligned ontology $O_u$ (i.e., $O_u = O_1 \cup O_2 \cup M$) must not introduce new subsumption relationships between concepts within the input ontologies. This variant follows the assumption of disjointness proposed in (Schlobach, 2005). So if two atomic concepts from one of the input ontologies are not involved in a subsumption relationship nor share a common sub-concept (excluding $\perp$), they can be considered as disjoint. Hence, if the input ontologies are extended with sufficient disjointness axioms, then the problem of detecting conservativity violations is reduced to an alignment incoherence detection. This detection is done in the same way as LogMap[42] (Jiménez-Ruiz & Cuenca Grau, 2011) which applies the following steps : (*i*) Extraction for each of the two input ontologies a locality-based module, as proposed in (Cuenca Grau et al., 2008), using only the entities involved in the alignment *M*. (*ii*) Encoding the input mappings *M* as a set of propositional implications simultaneously with the classifications of both modules provided by an OWL2 reasoner as Horn propositional theories. These theories include rules of the form $A_1 \wedge \ldots \wedge A_n \rightarrow B$ for the concept hierarchy together with rules of the form $A_i \wedge A_j \rightarrow \mathit{false}$ for the explicit disjointness relationships between concepts. (*iii*) Structural indexing the aligned ontology $O_u$ using an interval labeling schema (Agrawal et al., 1989), in order to avoid the reuse of a logical reasoner by storing directed acyclic graphs. The indexing allows to answer many entailment queries over the concept hierarchy as an index lookup operation. (*iv*) Reducing the conservativity problem to an alignment incoherence detection following the notion of assumption of disjointness (Schlobach, 2005). This is possible thanks to automatic addition of sufficient disjointness axioms into each module and detecting the set of axioms which leads to the unsatisfiability in the aligned ontology $O_u$. To achieve this, two techniques are proposed: (*i*) Exploiting only the structural indexation to check if two propositional variables are disjoint; they keep a sub/super-class relationship, or they share a common descendant, in order to add as many disjointness rules as possible, which is prohibitive for large ontologies. (*ii*) Exploiting both structural indexation and classification of the aligned ontology $O_u$ provided by an OWL2 reasoner in order to focus on the cases where a conservativity principle violation occurs in this ontology, and dealing only with the relative propositional variables. In both techniques, the structural index is updated to take into account the new disjointness rules. Since the

[42] http://krrwebtools.cs.ox.ac.uk/logmap/

conservativity problem is reduced to an alignment inconsistency problem, in (Solimando et al., 2014a), a previous work of (Solimando et al., 2016), the reparation of conservativity violations is also done in the same way as LogMap (Jiménez-Ruiz & Cuenca Grau, 2011) to repair consistency violations. The iterative alignment repair process checks for every propositional variable $A \in P_1^d \cup P_2^d$, the satisfiability of propositional theory $P_A = P_1^d \cup P_2^d \cup M \cup \{true \rightarrow A\}$. In the case of unsatisfiability, the algorithm allows to record conflicting mappings involved in the unsatisfiability, which will be considered for the subsequent repair process. Here too, the unsatisfiability will be fixed by removing some of the identified mappings using the correspondence confidence value as a differentiating factor. In the scenarios where the confidence of the mapping is missing (*e.g.*, in reference or manually created mapping sets) or unreliable, this mapping repair technique computes fresh confidence values based on the locality principle cited in (Jiménez-Ruiz et al., 2011).

In (Solimando et al., 2014b), further work is added to the works in (Solimando et al., 2014a). It is about *CycleBreaker* algorithm (called *EqRepair* Algorithm in (Solimando et al., 2016)). This algorithm is designed to detect another variant of conservativity violation called equivalence violation at a taxonomic level and calculates a minimal repair by a logic program. The equivalence violation is about a sets of correspondences that form a cycle of type: $e_1 \rightarrow e_2$ ; $e_2 \rightarrow e'_2$ ; $e'_2 \rightarrow e'_1$ ; $e'_1 \rightarrow e_1$ , such as, $(e_1, e'_1) \in O_1$ and $(e_2, e'_2) \in O_2$. This makes any entity reachable by starting from any other entity. Therefore, the equivalence relation necessarily replaces the subsumption relation. The equivalence violations detection process calculates the aligned ontology $O_u$ and constructs its graphical representation, using its named concepts as the set of vertices, and the subsumption axioms between these concepts as weighted arcs. The detection of cycles for a graph via Tarjan[43] algorithm (Tarjan, 1972) can be reduced to the calculation of all its *strongly connected components* (SCCs). Not all detected cycles lead to equivalence violations; there are two types of cycles: *unsafe* cycles and *safe* cycles, to distinguish between those producing a violation or not. The goal then is to detect all unsafe cycles, by simply detecting the set of SCCs containing at least one of the two projections on the input ontologies which is not a local SCC. This repair program uses the weighting in the arcs of unsafe cycles to eliminate the arc with the lowest weight. This elimination does not concern all unsafe cycles, since there

[43] The Tarjan algorithm takes as input a directed graph and returns a partition of the vertices of the graph corresponding to its strongly connected components.

are common arcs in several cycles, and their removal ensures the minimality of change principle by computing a diagnosis which is the set of arcs, once removed, cracks all unsafe cycles. Similarly to (Jiménez-Ruiz et al., 2011), the work in (Solimando et al., 2016) which assembles (Solimando et al., 2014a) and (Solimando et al., 2014b) works can serve as a solution to the alignment conservativity under ontology change problem. However, it considers only the subsumption axioms as conservativity violations (the equivalence violation is treated as a two-way subsumption), while our approach doesn't depend on the ontological axiom type.

It is also clear that this type of technique wastes all the effort provided before the ontological change. What cause the emergence of methods aiming to adapt the alignment following ontological changes instead of calculating a new one from scratch. Also according to Figure 3.3, we explore in the next section the right branch of the classification tree, to examine alignment evolution methods under ontology change in the alignment adaptation context.

### 3.3.2 Alignment adaptation methods

Starting from the idea that alignment calculation is not a trivial task, and since change in ontologies may trigger change in alignment, a set of approaches has emerged to deal with the alignment adaptation problem. In this context, alignment evolution methods aim to reuse as much as possible the old alignment by adapting it to the ontological change. Similar to the ontology matching context, the adaptive methods differ according to the performance targeted in the outputting alignments as shown in Figure 3.3. There are three different levels for this kind of method: structural consistency methods, logical consistency methods and conservativity methods.

#### 3.3.2.1 Systems dealing with structural consistency

A first set of approaches aims to guarantee structural consistency. For example, in the work of Khattak et al. (2015), the ontology is considered as a directed acyclic graph (DAG). The authors explore the change history log (Khattak et al., 2008) of the evolved ontologies and delete all correspondences concerned by the change. Then, they add new correspondences by partially re-computing the alignment and reuse completely its unaffected part. The changed elements in the evolved input ontology are automatically matched with the complete current version of the other ontology. Although this approach

reduces significantly the time required to maintain alignment compared to the time spent when alignments are fully re-computed from scratch using ontology matching tools, it doesn't much profit from the availability of the ontology change to adapt alignment. Instead, the approach uses changes only for filtering affected correspondences. Seeking new match for changed entities can only ensure a structural consistency of the alignment but not the logical consistency.

In (Martins & Silva, 2009), an alignment is an instance of Semantic Bridge Ontology (SBO). This ontology represents a set of semantic correspondences between input ontologies. When concepts are deleted from one of the input ontologies, the alignment evolution process tries to preserve the semantics of SBO by detecting and correcting invalid entities in it. This reduces the consistency violation detection to a concept satisfaction problem. Consequently, the authors propose two methods to correct invalid entities of SBO inspired by ontology evolution strategies (Stojanovic, 2004). The first method is user driven alignment evolution. The user chooses the strategy, and the system automatically takes care of the consequences of the changes following the execution of the chosen development strategy. In the second method, the system uses a change log to predict the ontology evolution strategies. This log stores the exact sequence of the performed changes to update the ontology. Thereafter, a list of rules is used to identify the scenario of the ontology evolution. This scenario determines the alignment evolution strategy. In this approach, only deleted concepts are considered. Thus, the approach correctly handles the violation of the alignment structural consistency.

To adapt an alignment to changes in input ontologies, Groß et al. ( 2013) are based on the composition of the old alignment with some generated alignment between versions of the evolved ontology. The alignment composition adapts the old alignment relying on the composition of the set-theoretic relations and uses some functions, such as the maximum or the aggregation, to combine their associated semantics similarities. The authors present two approaches: the composition-based and diff-based adaptation approaches. Using the ontology matching tool GOMMA (Kirsten et al., 2011), the composition based approach converts the implicit ontology change represented by the presence of versions to an alignment. The diff-based approach uses COnto-Diff tool (Hartung et al., 2013) to identify changes between evolved ontology versions, then converts every type of change to a semantic relation between entities concerned by this change. Both approaches seek new

match for added concepts with concepts of the target ontology to enhance the alignment with new correspondences. We have implemented the composition of an alignment between ontology versions with an initial alignment in a previous work (Atig et al., 2013), and we can testify on the simplicity of its implementation, but nothing guarantees that resulting alignment is valid. Moreover, the correctness of this operation depends on the correctness of the composed alignments. Both proposals rely on heuristic rules to generate an alignment between versions. Thereby, no guarantees are given to ensure the alignment validity and the logical consistency. Furthermore, the alignment composition is an incomplete method which might lead to unnecessary missing of some correspondences in the new alignment.

The work in (Dos Reis et al., 2013) aims to automatically adapt the affected alignment correspondences according to the type of the ontological change. A change handler is proposed to convert every ontological change to a set of mapping adaptation actions. Based on the same tool COnto-Diff as the previous approach, this one compare versions and categorize changes according to a revision change, an addition change or a deletion change. The authors proposed two types of actions: (*i*) two atomic actions : represented by correspondences addition, correspondences remove, and (*ii*) three composed actions: represented by correspondences move, correspondences derivation, and modification of semantic relations. While, the move action re-allocates a correspondence in the alignment when it is judged invalid, and the derivation action creates a modified copy of a correspondence which is still considered as valid, the modification action is used in conjunction with these two actions to change the type of semantic relations. Before every mapping adaptation action, an operation of matching is performed to determine the position (e.g, the concept) where the new correspondence should be re-allocated or from which is derived. The change handler associates an action or more to every type of change. The move action is associated to a revision change or to a deletion change, while the derivation action is associated to an addition change. Similarly to the previous approaches, the alignment validity is not explicitly defined. Furthermore, the move and derivation actions rely on matching operations. Consequently, it is not clear how the approach can ensure the alignment validity relying only on performing such mapping adaptation actions.

#### 3.3.2.2 Systems dealing with logical consistency

Euzenat considers his work in (Euzenat, 2015) as a first step to understand revision problems in networks of ontologies. This network is formed by a set of ontologies connected by a set of alignments. He studies the problem of restoring consistency in a such network when concerned ontologies were evolved or the alignment was improved by adding new correspondences. Inconsistency may manifest in two ways: local inconsistencies or a global inconsistency. A local inconsistency is an ontology inconsistency or an alignment inconsistency, while global inconsistency arises in the network, but ontologies and alignments are consistent in isolation. According to the author, the local revision of the concerned ontology or the concerned alignment is the only possible solution for local inconsistencies and these local operations can be used independently to resolve the global inconsistency. The approach provides alternative strategies in order to minimize the network change. For instance, one can only change the concerned ontology, while others can change only alignments, since ontologies are the pillar of knowledge, and it's worth do not modify them only if there is not another way. Although global inconsistency revision can appear as an adaptive operation, it is logically considered in the ontology matching context, since it is based on the local revision to correct global inconsistencies. Furthermore, the presented framework lacks practicability, since closed sets are infinite or at least very large sets that cannot be incorporated easily into a computational framework (Peppas, 2008).

For detecting ontology change preservation violations, Zahaf considers in (Zahaf, 2012) that alignment between ontology versions cannot be considered as consistent, since some correspondences propagate axioms from one version to another. This violates the constraint of conserving the changed meaning. His goal is to identify these correspondences and provide means to choose among them which must be eliminated. The identification of these correspondences first involves constructing a change log between ontology versions. This log is simply obtained by identifying the signature of the propagated axiom. An axiom in a version is considered as persistent if the other version contains its image. The image of an axiom is obtained by systematically replacing signature elements of this axiom by their correspondents according to the alignment. To repair ontology change preservation violations, the author introduces an order relation called relevance relation on the signature elements of the propagated axiom, which compares the degrees of intentional persistence of these elements. The intentional persistence of an element signature is expressed as the ratio of the number of occurrences

of this element in the set of persistent axioms for a given version on the total number of persistent axioms. The signature element that has the less intentional persistent allows to choose the correspondence to be eliminated from the initial alignment. In the equality case, the choice is left to the user. This approach provides a foundation for future works of the same author. Zahaf & Malki (2016) are inspired by belief base revision theory to define a formal framework in order to preserve the ontological change meaning and ensure a consistent alignment evolution under ontology change. The framework includes two possible operations, namely: (*i*) alignment revision which restores the consistency following adding new axioms in input ontologies, and (*ii*) alignment contraction which ensures not entailing again the removed axioms. The authors draw a set of constraints that an alignment evolution under ontology change should satisfy in order to be correctly evolved. Then, based on diagnostics theory, they propose an automatic method to reach this objective. A conflict set of correspondences responsible for every violation is calculated according to the two defined operations (revision and contraction). Finally, a diagnosis is computed and discarded from alignment. Despite the difference in the violations nature between consistency and conservativity principles, this approach is similar to ours in terms of violation detection process for removed axioms (alignment contraction). However, it is completely different in the case of added axioms (alignment revision).

#### 3.3.2.3 Systems dealing with conservativity principle

To our knowledge, the problem of Alignment Conservativity Under Ontology Change has not been studied yet. So, according to the proposed categorization for alignment evolution methods as shown in Figure 3.3, we believe that the current work is the first to address this problem. Therefore, while waiting for other approaches to emerge in the same context, we consider this proposal a first step to perfect the task of alignment evolution following the change in the input ontologies.

## 3.4 Conclusion

To conclude this analysis, we note that approximately in literature, all alignment repair systems adopt a common principle which suggests that the input ontologies are immune during the repair phase, except the previously discussed ontology local inconsistency restoration in (Euzenat, 2015), and the work in (Pesquita et al., 2013) which proposes to

update the ontologies during the automatic calculation of repairs. Furthermore, we observe that alignment revision strategies in the studied approaches differ in the nature of violations and therefore the purpose of the repair process. On the other hand, most of them use the conflict set and diagnosis notions inspired by diagnosis theory. A conflict set is the set of correspondences responsible for the considered violations, while a diagnosis is the set of correspondences that have the lowest confidence values in each conflict set. We follow the same strategies in this work whether through immunization input ontologies in the repair process, or concerning conflict set and diagnosis.

# Chapter 4. Methods

## 4.1 Introduction

In chapter 3, we have positioned the alignment adaptation problem with respect to the conservativity principle following ontological changes against panoply of existing techniques. These techniques deal with the alignment evolution problem in both adaptation and matching calculation contexts. In this chapter, we unveil our proposal to remedy this problem. We propose at the beginning two patterns which allow to detect conservativity violations following either an addition or deletion of axioms in the input ontologies. We then present how to exploit the detection phase to adapt the original alignment to these ontological changes with regard to the conservativity principle. In what follows, we detail in Section 4.2 the detection process of alignment conservativity violations under ontology change. Section 4.3 shows the repair strategy of these violations. Then, we conclude the chapter in Section 4.4.

## 4.2. Detecting conservativity violations under ontology change

The conservativity principle as a deductive difference already suffers from two major drawbacks (Lutz et al., 2007; Lutz & Wolter, 2010), namely: (*i*) the lack of algorithm available for computing deductive difference for DL logics, and (*ii*) the massive, up to infinite, number of entailments in this difference. In order to avoid these drawbacks, we suggest an approximation of the deductive difference in the context of alignment evolution under ontology change.

In this dissertation, we only consider alignments with the equivalence relations. This is not a disadvantage of our approach because it is always possible to find a subset of the alignment with only equivalent relations. An equivalence relation expresses that linked entities represent the same thing in the domain of discourse. In this case, the alignment constitutes an isomorphism of ontological signature (Kalfoglou & Schorlemmer, 2003) connecting the vocabulary of two ontologies so that the axioms specifying the linked

entities are preserved or conserved. This conservativeness must remain valid throughout the ontologies life cycle. Otherwise, we must register conservativity violations.

**Definition 4.1** (Conservativity Violations Under Ontology Change Detection Patterns). Let $O_i$ be an ontology which has evolved to a new version $O_{i2}$ and $M$ an alignment between two ontologies $O_i$ and $O_j$ might manifests conservativity violations under ontology change if and only if:

- *For all added axiom* $\delta^+$ *such that* $sig(\delta^+) \subseteq Q(O_{i2})$, *we have* $O_{i2} \cup_M O_j \models M(\delta^+)$ *but* $O_j \not\models M(\delta^+)$
- *For all deleted axiom* $\delta^-$ *such that* $sig(\delta^-) \subseteq Q(O_{i2})$, *we have* $O_{i2} \cup_M O_j \models M(\delta^-)$

Note that violations of the conservativity only concern axioms whose signature is fully involved in the alignment, which means that the signature elements of any axiom are matchable entities. The following Example 19 illustrates a situation in which the images of the added axioms in $O_{i2}$must exist as a logical consequence of the ontology $O_j$.

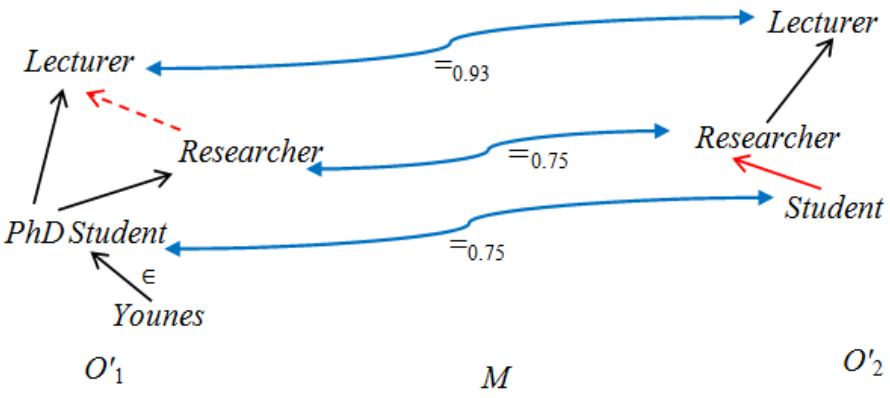


**Figure 4.1. Evolution of ontology $O_2$ into new version $O_2'$**

***Example 19.*** *Following Example 18, assuming that the second input ontology* $O_2$ *has been evolved into* $O'_2$ *(solid red arrow in Figure 4.1), let* $O_2' = O_2 \cup \{2: Student \subseteq 2: Researcher\}$ *be a new version of* $O_2$. *The current change restores the conservativity of a subset of M (i.e.,* $\{1: PhDStudent =_{0.75} 2: Student; 1: Researcher =_{0.75} 2: Researcher\}$*), since that,* $O'_1 \models M(2: Student \subseteq 2: Researcher) = \{1: PhDStudent \subseteq 1: Researcher\}$. *However, M is not fully conservative, since that,* $O'_1 \not\models M(2: Researcher \subseteq 2: Lecturer) = \{1: Researcher \subseteq 1: Lecturer\}$*represented by a dashed red arrow in Figure 4.1.*

Considering now the second situation. As illustrated in Example 20, the images of the deleted axioms from $O_{i1}$ must not exist as a logical consequence of the ontology $O_j$. There also, the signature of these deleted axioms must be included in the set of the ontology $O_{i2}$ matchable entities.

***Example 20.*** *Following Example 19, assuming now that ontology $O'_1$ has been evolved once again into $O''_1$; let $O''_1 = O'_1 / \{1: PhD\ Student \sqsubseteq 1: Researcher\}$ be a new version of $O'_1$. The current change breaks the conservativity of the subset $\{1: PhDStudent =_{0.75} 2: Student; 1: Researcher =_{0.75} 2: Researcher\}$ of M, since that,$O'_2 \models M(1: PhDStudent \sqsubseteq 1: Researcher) = \{2: Student \sqsubseteq 2: Researcher\}$.*

## 4.3 Reparation of Conservativity Violations Under Ontology Change

Several alternatives can be considered to adapt an alignment under ontology change with regard to the conservativity principle. One of them is the empty alignment where we discard all its correspondences. It is evident that the empty alignment respects the conservativity principle since it doesn't connect any entities. Consequently, the aligned ontology is formed only by the fusion of the input ontologies and no knowledge propagation is expected. Nevertheless, the empty alignment doesn't make any sense from practical point of view, and we need to compute the new alignment from scratch. According to the principle of minimal change (Peppas, 2008), an ideal solution would be to change only the relevant correspondences that cause problems. Furthermore, the consensual property targeted by the alignment and all problems relating to its calculation (Euzenat & Shvaiko, 2013) stab in the usefulness of this strategy. A second alternative would be to correct conservativity violations without considering the ontological change (Solimando et al., 2016). Since this strategy is essentially based on the exhaustive analysis of alignment correspondences jointly with all axioms of the two input ontologies to detect violations, it greatly influences the speed of the repairs computation time, especially in the case of tiny ontological changes.

In this dissertation, we adopt a simple and efficient reparation approach, which consists in correcting alignment, while respecting its conservativity upon a change in its related ontologies. In other words, this approach aims to give means to choose among alignment correspondences which of them must be eliminated to remedy conservativity violations following the occurred ontological changes. In order to preserve as much as possible the original alignment, the elimination should be minimal. For this reason, we adapt techniques from diagnosis theory to design this operation. The diagnosis theory presented for the first time in (Reiter, 1987) states that a diagnosis task is generally defined in terms of a set of components *COMP* in which a fault might have occurred, a behavior of a system defined by the system description *SD* and a set of observations *OBS* (also called symptoms). A diagnosis is defined as the minimal set $\Delta \subseteq COMP$ such that the

observations *OBS* are explained by a subset of the components having abnormal behavior. Based on an appropriate formalization of the concept of a conflict set, Reiter proposes in (Reiter, 1987) a method to compute diagnosis. A conflict set is a subset of the system components that together produce an abnormal behavior. Since the same symptom can be caused by different conflict sets, a diagnosis is defined as the minimal set which intersects each conflict set.

In the alignment conservativity under ontology change problem, we consider on the one hand, alignment correspondences to be the set of diagnosed components, and on the other hand, the aligned ontology to be the system description, while observations are provided in terms of conservativity violations. Correspondences are assumed to be abnormal if they cause alignment conservativity violation. A conflict set is a subset of correspondences that together cause conservativity violation. Conservativity violation under ontology change as presented above (see Definition 4.1) is either (*i*) a previously nonexistent image in one input ontology but entailed using the alignment in the case of axiom addition in the second input ontology, or (*ii*) a deleted axiom but regenerated in one input ontology using the alignment jointly with its image in the second input ontology. In what follows, $\delta$ represents the undesired axiom in both cases violating the conservativity principle.

In order to respect the minimal change principle (Peppas, 2008), so as not to fall into an complete removal thereby eliminating all correspondences of the conflict set, we present in definition 4.2 the notion of minimal conflict sets. The relevance in this notion is to repair the violation of conservativity principle by fixing only one element in each conflict set.

**Definition 4.2** (Minimal Conflict Set). Given two input ontologies, namely $O_{i1}$, $O_j$, and an alignment $M$ between them. Consider that ontology $O_{i1}$ has evolved to $O_{i2}$. In the case of conservativity principle violation following $O_{i1}$ evolution, a subset $C$ of the alignment $M$ is a minimal conflict set if and only if $C \subseteq M$ *and* :

- *For all added axiom* $\delta^+$ : $O_{i2} \cup_C O_j \models M(\delta^+)$ *and* $\forall C' \subseteq C$ *we have* $O_{i2} \cup_{C'} O_j \not\models M(\delta^+)$.
- *For all deleted axiom* $\delta^-$ : $O_{i2} \cup_C O_j \models \delta^-$ *and* $\forall C' \subseteq C$ *we have* $O_{i2} \cup_{C'} O_j \not\models \delta^-$.

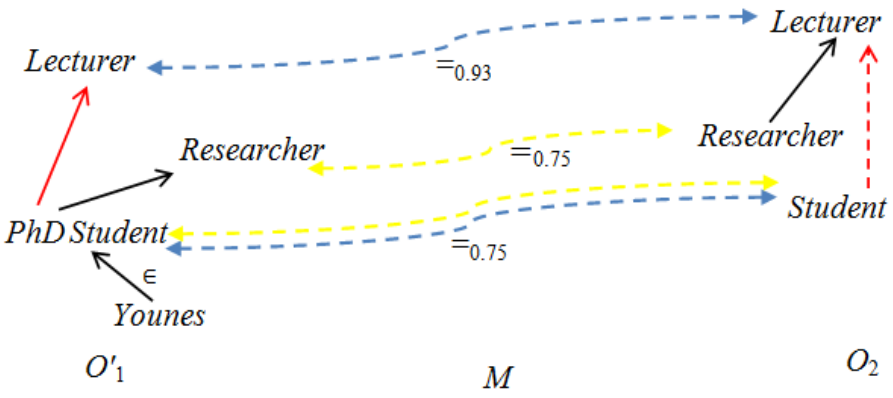


**Figure 4.2. Two conflict sets for a single conservativity violation**

It is quite simple to conceptualize an algorithm to find a conflict set as illustrated in Table 4.1. It is only required to removing each element of *M* and testing if the remaining ones still violates the conservativity principle. If this is not the case, the element is reintroduced in *M*. The result is a minimal conflict set, since it is a subset of *M* that implies $\delta$, and no subset of this result still implies it. This algorithm is an adaptation of that presented in (Zahaf, 2017) to calculate a minimal conflict set in the context of alignment consistency problem, while taking into account the difference in the nature of the violations and the undesired axiom type regarding the conservativity problem. Example 21 illustrates the progress of the algorithm.

**Table 4.1. Minimal conflict set algorithm**

| Algorithm 1: minimal conflict set |
|---|
| MinConflictSet (M,o1,o2,$\delta$)<br>Input : o1,o2 // two ontologies<br>M // M is an alignment between o1 and o2<br>$\delta$ // $\delta$ is an undesired axiom<br>Output : M // a minimal conflict set<br>1. for $c \in M$<br>2. do<br>3. if $O_1 \cup_M O_2 \models \delta$<br>4. then $M \leftarrow M \setminus \{c\}$<br>5. return $M$ |

***Example 21.*** *Following Example 18, we demonstrate now how to compute the minimal conflict sets upon revising one of the connected ontologies, by using the algorithm 1. Let* $O'_1 = O_1 \cup \{1: PhDStudent \sqsubseteq 1: Lecturer\}$*, and therefore, the axiom 2: Student* $\sqsubseteq$ *2: Lecturer will be an undesirable logical consequence* $\delta$ *in the ontology* $O_2$*.*

*1. The algorithm iterates over the elements of M (Line 1). Let's assume that it iterates from left to right.*

*2. For c = {1: Lecturer $=_{0.93}$ 2: Lecturer} (Line 1). Checks $O'_1 \cup_M O_2 \models \delta$ (line 3). So it removes 1: Lecturer $=_{0.93}$ 2: Lecturer from $\mathcal{M}$ (line 4).*

*3. For c = {1: Researcher $=_{0.75}$ 2: Researcher} (Line 1). Checks $O'_1 \cup_M O_2 \not\models \delta$ (line 3). Then it does not change $\mathcal{M}$ (line 3).*

*4. For c = {1: Phd Student $=_{0.75}$ 2: Student} (Line 1). Checks $O'_1 \cup_M O_2 \not\models \delta$ (line 3). Then it does not change $\mathcal{M}$ (line 3).*

*5. Return $\mathcal{M}$ = {1: Phd Student $=_{0.75}$ 2: Student; 1: Researcher $=_{0.75}$ 2: Researcher} which is a minimal conflict set (line 5) (also shown by the dashed yellow arrows in Figure 4.2).*

*6. The algorithm iterates once again over the elements of the original M (Line 1), but from another starting point this time.*

*7. For c = {1: Researcher $=_{0.75}$ 2: Researcher} (Line 1). Checks $O'_1 \cup_M O_2 \models \delta$ (line 3). So it removes 1: Researcher $=_{0.75}$ 2: Researcher from $\mathcal{M}$ (line 4).*

*8. For c = {1: Phd Student $=_{0.75}$ 2: Student} (Line 1). Checks $O'_1 \cup_M O_2 \not\models \delta$ (line 3). Then, it does not change $\mathcal{M}$ (line 3).*

*9. For c = {1: Lecturer $=_{0.93}$ 2: Lecturer} (Line 1). Checks $O'_1 \cup_M O_2 \not\models \delta$ (line 3). Then, it does not change $\mathcal{M}$ (line 3).*

*10. Return $\mathcal{M}$ = {1: Phd Student $=_{0.75}$ 2: Student; 1: Lecturer $=_{0.93}$ 2: Lecturer} which is another minimal conflict set (line 5) (also shown by the dashed blue arrows in Figure 4.2).*

*11. The algorithm iterates once again over the elements of the original M (Line 1), but from a third point this time.*

*12. For c = {1: Phd Student $=_{0.75}$ 2: Student} (Line 1). Checks $O'_1 \cup_M O_2 \not\models \delta$ (line 3). Then it does not change M (line 3).*

*13. For c = {1: Lecturer $=_{0.93}$ 2: Lecturer} (Line 1). Checks $O'_1 \cup_M O_2 \models \delta$ (line 3). So it removes 1: Lecturer $=_{0.93}$ 2: Lecturer from M (line 4).*

*14. For c = {1: Researcher $=_{0.75}$ 2: Researcher} (Line 1). Checks $O'_1 \cup_M O_2 \not\models \delta$ (line 3). Then, it does not change M (line 3).*

*15. Return M = {1: Phd Student $=_{0.75}$ 2: Student; 1: Researcher $=_{0.75}$ 2: Researcher} which is the same first minimal conflict set (line 5).*

*As a result, the conflict sets responsible of the conservativity violation of the alignment M upon $O'_1 = O_1 \cup$ {1: PhDStudent $\sqsubseteq$ 1 : Lecturer}, are:*

*$C_1$ = {1: Phd Student $=_{0.75}$ 2: Student; 1: Researcher $=_{0.75}$ 2: Researcher}(dashed yellow arrows in Figure 4.2).*
*$C_2$ = {1: Phd Student $=_{0.75}$ 2: Student; 1: Lecturer $=_{0.93}$ 2: Lecturer}(dashed blue arrows in Figure 4.2).*

A diagnosis is known to be the minimal set of correspondences which intersects each minimal conflict set (Meilicke & Stuckenschmidt, 2007). This intersection involves the set of correspondences with the intention of being eliminated to restore the alignment conservativity before changing in the related ontologies. Actually, this criterion of minimality is not always appropriate in alignment debugging problem, since it does not take into account the confidence value as factor of choice between the correspondences to be eliminated. For this purpose, we choose to penalize the least confident correspondence compared to the others. This choice is justified by the fact that the role from the start to incorporate a confidence value, is the rate of assurance that we put on each correspondence. Therefore, in order to choose among correspondences the one with the lowest confidence value, we introduce an order relation on alignment correspondences based on these values. The correspondence with the lowest confidence value in each conflict set represents an element of the diagnosis set.

**Definition 4.3** (Alignment Diagnosis). Given *MC* a set of minimal conflict sets of an alignment *M* with regard to an undesired axiom $\delta$ violating the conservativity principle, $\Delta$ is a diagnosis for the alignment *M* with respect to *MC* if and only if: $\Delta = \{c = (e, e', r, n) \mid C \in MC, c \in C, n = min\{n_i \mid (e_i, e'_i, r_i, n_i) \in C\}\}$.

The algorithm for calculating alignment diagnosis presented in Table 4.2 is based on a condition stipulating that it is formed by taking one correspondence from each minimal conflict set, and this correspondence should have the less confidence value when compared with the others. Note that, algorithm 2 acts as the binary search algorithm[44]; if we group all minimal conflict sets in a one tree, such that nodes are labeled by minimal conflict sets and edges are labeled by the elements of these minimal conflict sets, this algorithm develops just one branch of the tree which corresponds to the correspondence with lowest confidence value in the generated conflict set. Hence, this algorithm runs in logarithmic time at worst. Example 22 and Figure 4.3 illustrate the progress of the algorithm.

[44] Binary search is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array; if they are unequal, the half in which the target cannot lie is eliminated and the search continues on the remaining half until it is successful or the remaining half is empty.

**Table 4.2. Binary search based alignment diagnosis algorithm**

| Algorithm 2: Binary search based alignment diagnosis |
|---|
| BinarySearchBasedAlignmentDiagnosis *(*M,o1,o2,*δ)*<br>Input : o1,o2 // two ontologies<br>M // M is an alignment between o1 and o2<br>*δ* // *δ* is an undesired axiom<br>Output : Δ // an alignment diagnosis<br>1. Δ ← ∅<br>2. while $O_1 \cup_M O_2 \models \delta$<br>3. do<br>4. *CS* ← MinConflictSet *(*M,o1,o2,*δ)*<br>5. *Clv* ← CorrespWithLowestConfidValue(*CS*)<br>6. Δ ← Δ ∪ {*Clv*}<br>7. *M* ← *M* \{*Clv*}<br>8. Return Δ |

***Example 22.*** *Following Example 21, we want to compute the diagnosis of the alignment M:*

*1. Algorithm 2 starts by computing a minimal conflict set. Let it the same as the first one in Example 21:*
*$CS_1$ = {1: Phd Student =$_{0.75}$ 2: Student; 1: Researcher =$_{0.75}$ 2: Researcher} (line 3-4).*

*2. Find the correspondence with the lowest confidence value in $CS_1$. Clv = {1: Phd Student =$_{0.75}$ 2: Student} (line 5).*

*3. Put Δ = Δ ∪{Clv} = {1: Phd Student =$_{0.75}$ 2: Student} (line 6).*

*4. Put M = M / {Clv} = {1: Lecturer =$_{0.93}$ 2: Lecturer; 1: Researcher =$_{0.75}$ 2: Researcher}(line 7).*

*5. Run algorithm 1 again, we obtain $CS_2$ = {1: Phd Student =$_{0.75}$ 2: Student; 1: Lecturer =$_{0.93}$ 2: Lecturer} (line 4)*

*6. Find the correspondence with the lowest confidence value in $CS_2$. Clv = {1: Phd Student =$_{0.75}$ 2: Student} (line 5).*

*7. Put Δ = Δ ∪{Clv} = {1: Phd Student =$_{0.75}$ 2: Student} (line 6).*

*8. Put M = M / {Clv} = {1: Lecturer =$_{0.93}$ 2: Lecturer; 1: Researcher =$_{0.75}$ 2: Researcher}(line 7).*

*9. Return Δ = {1: Phd Student =$_{0.75}$ 2: Student} which is a diagnosis of the alignment M (line 8).*

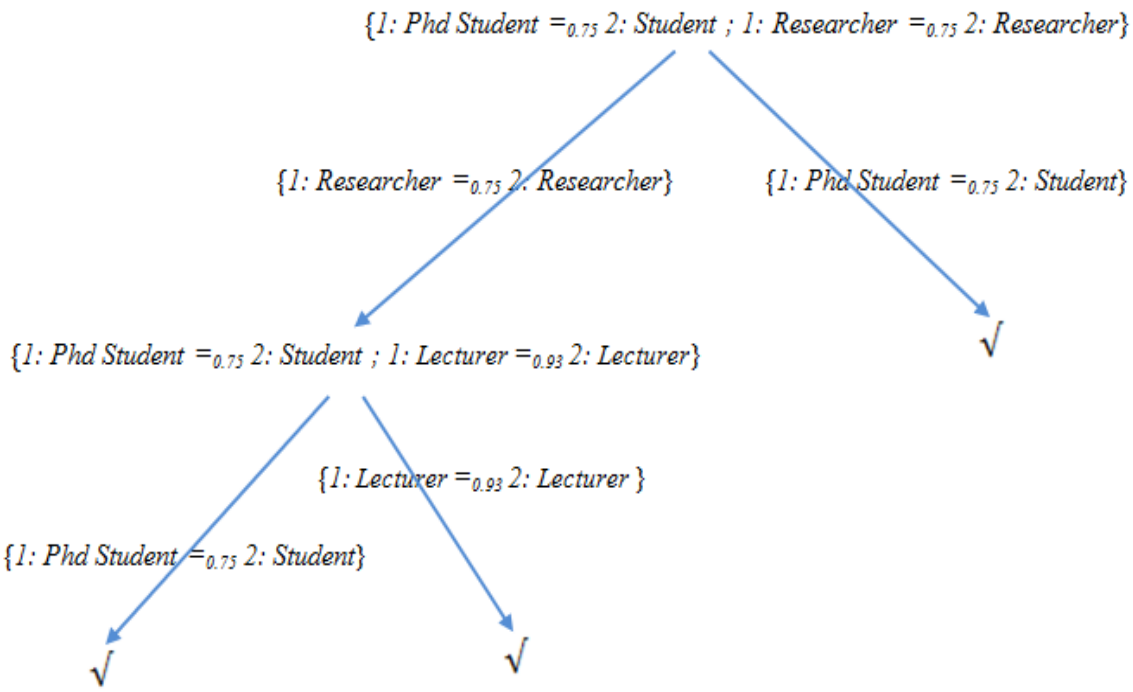


**Figure 4.3. Hitting set tree of the alignment *M* diagnosis**

Finally, the alignment repair process discards the diagnosis from the original alignment in order to restore its lost conservativity upon input ontologies evolution. The result of this revision is a repaired sub-alignment with respect to the conservativity principle.

**Definition 4.4** (Alignment Reparation). Given two input ontologies, namely $O_{i1}$, $O_j$, and an alignment $M$ between them. Consider that ontology $O_{i1}$ has evolved to $O_{i2}$, and subsequently generates a set of conservativity violations. $\Delta$ is the calculated diagnosis to restore the conservativity of the alignment $M$, and the result of this reparation is an alignment $M'$ such that, $M' = M \setminus \Delta$.

***Example 23.*** *Following Example 22, the repaired alignment M' by the obtained diagnosis* $\Delta$ *is:*

$M' = M \setminus \Delta = \{1: \text{Lecturer} =_{0.93} 2: \text{Lecturer} ; 1: 1{:}\text{Researcher} =_{0.75} 2{:}\text{Researcher}\}$

While the performance of the violations detection process depends mainly on the techniques used to produce syntactic difference between versions, the performance of alignment revision depends on the underlying representation languages of the ontologies. As illustrated in (Zahaf & Malki, 2018), the correctness of this operation needs monotonic and compact languages. The Monotony of a language of two ontologies $O$ and $O'$ stipulates

that, if $O \subseteq O'$ then $Closure(O) \subseteq Closure(O')$, while the Compactness specifies that, if $O \models \delta$ then, there is some subset $O \subseteq O'$ such that $O' \models \delta$. Fortunately, like OWL, such languages exist. The natural semantics of alignment respect the monotony and compactness criteria since it only extends ontologies by axioms expressed within the same language of ontologies. Following these conditions, the alignment revision satisfies the conservativity principle constraints. Indeed, a demonstration in (Zahaf & Malki, 2016) is that a diagnosis is a complete repair method, since it repairs all detected consistency violations. The same demonstration holds for conservativity principle by replacing the contradictory axioms by axioms violating conservativity.

As a final point, we can see that since confidence values incorporated in the original alignment correspondences, are calculated before ontology evolution and may be obsolete after that, it is not fair to rely on these values for the proposed alignment reparation process. Moreover, the minimal change is at stake for the alignment revision. Diagnosis based on confidence values criteria may lead to discard more correspondences than necessary. This could happen since some correspondences may have the same confidence value within a conflict set. Also, we can't restrict the order relation based on confidence values to be total. This is not realistic since we have no means to oblige ontology matching to generate such alignments. In general, we do not consider our approach as a turnkey method for alignment evolution, but rather a complement for this kind of approach, dealing with the conservativity violations upon ontological change.

## 4.4 Conclusion

In this chapter, we have presented our proposal to deal with the alignment adaptation problem with respect to the conservativity principle following ontological changes. We have dealt with two aspects: detecting conservativity violations and correcting them. In the aspect of detecting the conservativity violations, we have proposed two patterns. The first pattern serves to detect conservativity violations generated following the addition of new axioms within the input ontologies. While the second pattern serves to detect conservativity violations generated following the removal of axioms from the input ontologies. In the second aspect, we adapt an alignment repair method proposed in (Zahaf & Malki, 2016) in the context of alignment evolution consistency to repair conservativity violations in the context of alignment evolution under ontology change. This method is a diagnosis task inspired by the diagnosis theory (Reiter, 1987) that aims to compute and

eliminate from the alignment a subset of correspondences called a diagnosis, to fix the conservativity violation.

# Chapter 5: Implementation and Experimentation

## 5.1 Introduction

In Chapter 4, we have presented the theoretical framework of our contribution. In this chapter, we evaluate the feasibility of using our method to detect and correct conservativity principle violations under ontology change in practice. We present at the beginning (in Section 5.2) the environment of implementation of our method. In fact, the proposed approach, in this dissertation, is part of a larger project[45] to deal with the Ontology Alignment Revision problem. This allowed us to build on top and to extend an already existing alignment evolution platform to address ontology alignment conservativity violations under ontological change. The Section 5.3 describes the conducted experimentation to test our proposal applicability. To achieve the objective of this experiment, we present the used dataset, the accuracy measures, the steps of the experiment, the obtained results and the discussion of findings. Finally, Section 5.4 concludes the chapter.

## 5.2 Implementation

In chapter 3, we have distinguished two classes of alignment evolution methods. While methods of the former, called alignment adaptation methods, reuse as much as possible the old alignment, methods of the latter fit under the ontology matching context, compute from scratch the new alignment. According to Zahaf & Malki (2016), none of the approaches of both classes guarantee the preservation of the ontology change in alignment evolution task. The preservation of the ontology change is a special case of the conservativity principle which only concerns deleted axioms. Through this experience, we will test some methods to consolidate this argument by extending it to all cases of the conservativity principle. Mainly, the selected methods rely on ontology matching techniques for evolving

[45] https://www.researchgate.net/project/Ontology-alignment-revision#projectLog

alignments. Besides, they embed debugging techniques to diagnosis alignments for eventual consistency problems. By selecting these methods we want to show that neither ontology matching nor alignment debugging methods fit well for the problem of conservativity violations in the context of ontology alignment evolution under ontology change.

We built on top and extend the alignment repair platform presented in (Zahaf, 2017) to address the ontology alignment conservativity violations under ontological change. The platform embeds the OWL-API (Horridge & Bechhofer, 2009) and Alignment-API (David et al., 2011) libraries as a baseline for managing OWL ontologies and alignments between ontologies. Figure 5.1 illustrates the platform architecture. In what follows, we detail the components of this platform one by one.

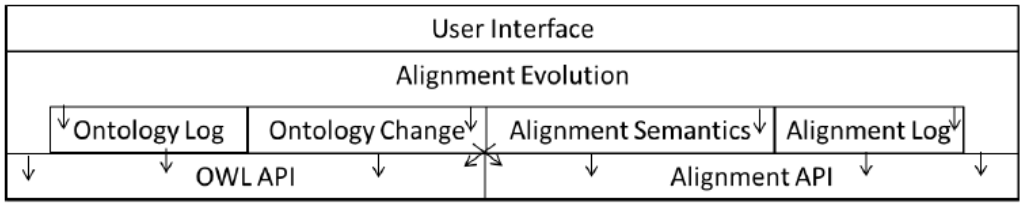


**Figure 5.1. Architecture of the alignment evolution system (Zahaf, 2017)**

### 5.2.1 OWL API

The OWL API is a high level Application Programming Interface (API) for working with OWL2 ontologies (Horridge & Bechhofer, 2011). Although the model explicitly supports the recent OWL2 Recommendation[46]. It also supports parsing and rendering in the syntaxes defined in the W3C specification (Functional Syntax, RDF/XML, OWL/XML and Manchester OWL Syntax) and other syntaxes, such as OBO flat file format[47] and KRSS Syntax[48]. The manipulation of ontological structures and the use of reasoning engines are also supported by the OWL API. Moreover, this API allows to import closure of ontologies written in different syntaxes. Figure 5.2 shows the main classes of the OWL API.

[46] https://www.w3.org/TR/2012/REC-owl2-primer-20121211/
[47] http://www.geneontology.org/faq/what-obo-file-format
[48] http://dl.kr.org/krss-spec.ps

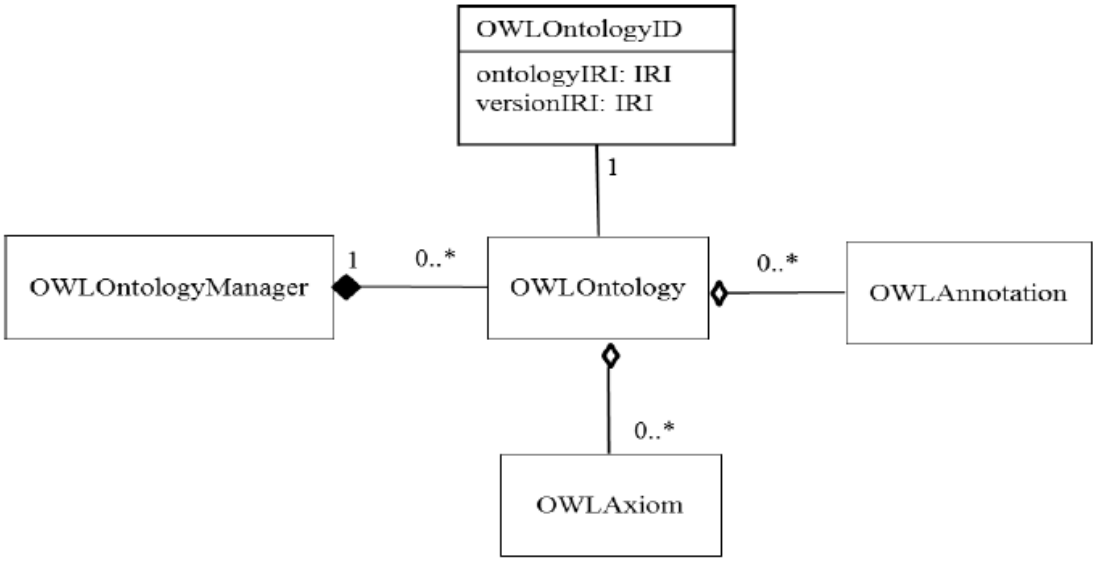


**Figure 5.2. UML diagram showing ontology management by the OWL API (Horridge & Bechhofer, 2011)**

*Figure 5.2 states that an ontology written in OWL is a set of OWL axioms. The axioms contained in an ontology are accessed through the OWLOntology interface. The OWLOntologyManager provides methods for creating, loading, changing and saving ontologies. The OWLOntology interface is considered as a superclass of these ontologies. In addition to all that, different tasks, such as consistency checking, computation of class / property hierarchies and axioms entailment, are supported by this API.*

### 5.2.2 Alignment API

The alignment API in turn is an API designed for managing alignments. It provides definitions of a set of Java interfaces and their basic implementations (David et al., 2011). Alignment API supports a set of tasks for manipulating alignments such as adding correspondences to alignments and deleting correspondences under a confidence threshold. These methods are provided through a set of representational classes shown in Figure 5.3.

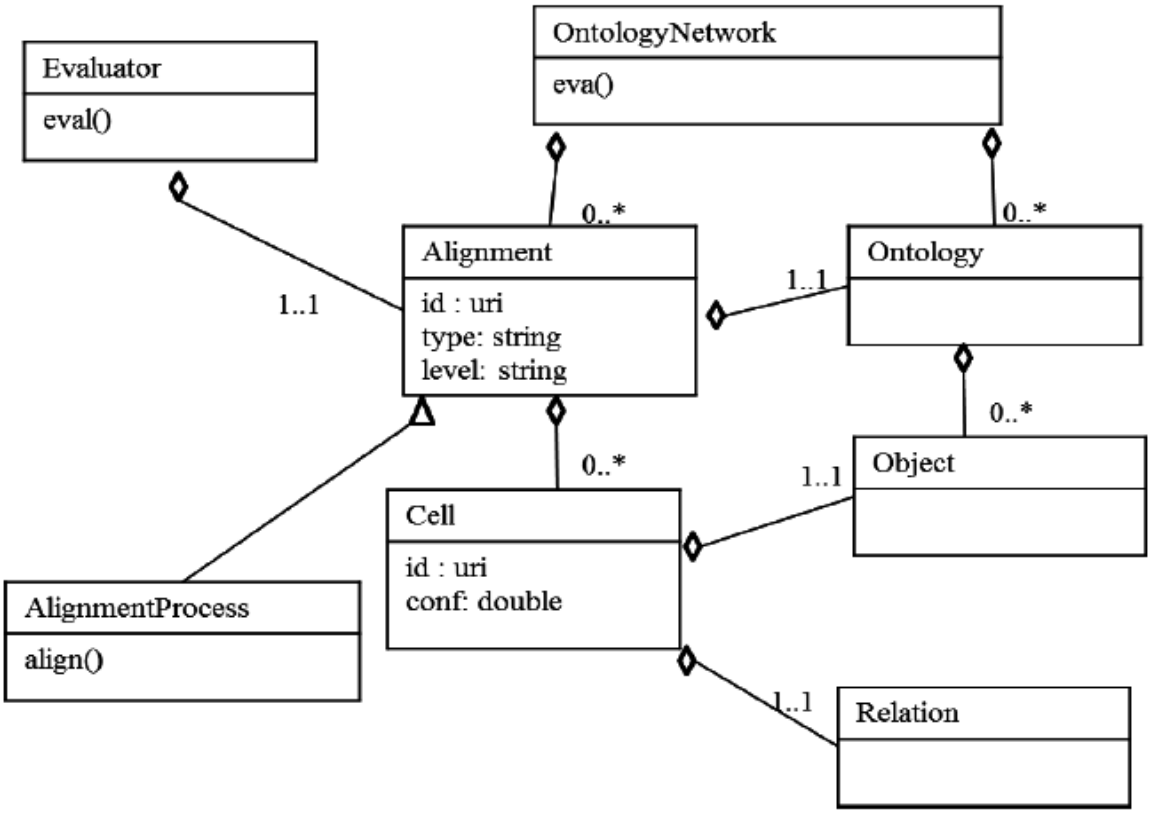


**Figure 5.3. UML diagram showing alignment management by the Alignment API (David et al., 2011)**

The Alignment class defines an alignment as a set of Cells. A Cell defines a Relation between two ontological entities. Besides, the class Cell supports any type of additional metadata including confidence values. Alignments and aligned ontologies form together a container which is represented by the OntologyNetwork class in the Alignment API. In addition, the Alignment API defines others classes for creating and evaluating alignments. AlignmentProcess provides a minimal processing structure for matching ontologies in order to create alignments. Evaluator provides methods for evaluating alignments by comparing a first alignment which may be taken as a reference and a second alignment.

### 5.2.3 Alignment evolution system

The alignment evolution system embeds the OWL API and the Alignment API libraries as a baseline for loading ontologies and for loading, modifying and storing alignments. In what follows, we present the different components of the system and the interaction between them:

- ***Ontology change*** component is responsible for identifying and representing the ontology change.
- ***Alignment log*** component embeds services for representing, storing and tracking the alignment change.
- ***Alignment evolution*** component implements the alignment evolution under ontology change repair.
- ***Alignment semantics*** component relies on the state-of-the-art of reasoners to check alignment consistency and entailments.

In the current work, we have extended the alignment evolution component to implement the method for detecting conservativity violations following the change in the input ontologies (see Chapter 4). By considering the alignment as an isomorphism (see Definition 2.7), our method checks the entailment of the image in one ontology of the axiomatic change in the other ontology. The next section allows to experiment the feasibility of using our method to detect and correct conservativity principle violations under ontology change in practice.

## 5.3 Experimental Evaluation

In the current part we first describe in Section 5.3.1, the dataset used for the experimentation. The dataset is made up of two parts. The first part concerns the used ontologies and the changes applied to them, while the second part concerns the alignments to be repaired. Although we adopt the same set of tests as in (Zahaf & Malki, 2016), we extend this dataset to include another test. We then present, in Section 5.3.2, the accuracy measures which will allow us to give an overview on strengths and weaknesses of the evolution methods used in this experimentation. Furthermore, a set on ontology matching tools needed in this experimentation are also presented in Section 5.3.3. Finally, the processes of the experiment itself and the obtained results are presented in detail in Section 5.3.4.

### 5.3.1 Dataset

#### 5.3.1.1 Ontologies and Change

OAEI[49], a coordinated international initiative, carries out annual campaigns for the evaluation of ontology matching tools. It uses a benchmark dataset for identifying strengths and weaknesses of matching systems. The benchmark dataset consists of a large set of artificial tests. These tests alter an initial ontology about the topic of scientific publications, and the task is to match it to the modified ontology. Modifications consist of inserting or deleting some features, e.g., replacing by random labels, deleting or inserting classes in the hierarchy, etc. The ontologies are described in OWL-DL and serialized in RDF/XML format. The initial ontology is that of test #101. It contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals.

We adapted a subset of the systematic benchmark for evaluating alignments evolution methods under ontology change. In what concerns ontological changes, we rearrange tests #101, #103, #104, #203, #223, #230 and #233, to form the new tests # 101-103-104, #101-203-223 and #101-230-233, according to the assessment requirements. We also consider ontologies 104, 223 and 233 as a version of 103, 203 and 230, respectively.

To generate the ontological change, we have used the method developed in (Zahaf, 2012) to compute the difference between versions. This method considers the ontological change operation as the set theoretical difference between signatures and axioms, respectively. Since the conservativity principle is a logical property which might concern only axioms whose signature is fully implied in alignments, we only consider the axiomatic change of matchable signatures. Table 5.1 summarizes the obtained change.

**Table 5.1. Ontological change between versions of the dataset**

| Difference Versions | Added Axioms | Deleted Axioms |
|---|---|---|
| 103-104 | 0 | 11 |
| 203-223 | 1 | 9 |
| 230-233 | 0 | 220 |
| 230-238 | 182 | 71 |

[49] http://oaei.ontologymatching.org/

The axioms removed from 103 compared to 104 are domains for object and data-properties. Besides adding new entities and related axioms to version 223, definitions of other entities have changed by adding axioms. The same holds for definitions of some entities in version 203 by removing axioms. Removed axioms are domains, ranges and some restrictions on properties. Since both do not have hierarchies, no axioms added between 230 and 233. Deleted axioms are due to the removal of object and data-properties. As the ontological change generated is mainly of suppression type, we extend the set with the additional test #101-230-238 to enrich it with addition type. Comparison between the versions 230 and 238 shows the removal of instance and related axioms, and adding other entities and axioms.

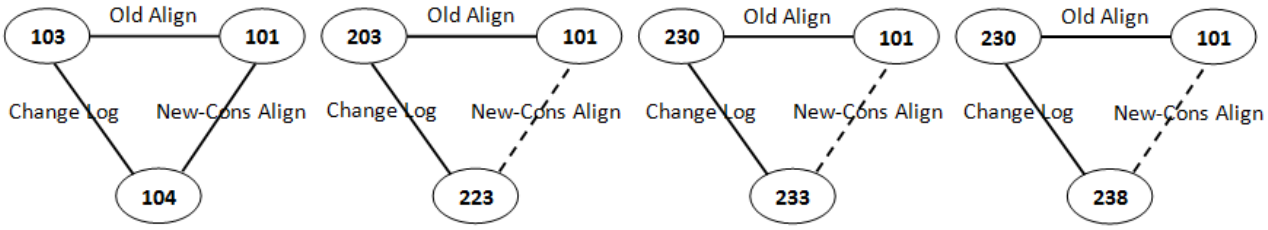


**Figure 5.4. Dataset**

#### 5.3.1.2 Alignments

Concerning alignments to be repaired, we consider as old alignments, those between the following ontologies pairs: 101-103, 101-203 and 101-230, while the alignments between the following ontologies pairs: 101-104, 101-223, 101-233 and 101-238 are the evolved alignments after change. Figure 5.4 schematizes this dataset.

### 5.3.2 Accuracy Measures

The considered dataset does not contain reference alignments to measure accuracy with respect to conservativity principle, which restricts the use of traditional precision methods. Therefore, to compare the performances of evolution methods in ontology matching context, we use the number of conservativity violations by changed axioms. In addition, we compare the elapsed time, as well as the rate of violations reparation for all methods. The violations reparation rate of an alignment $M$ is defined by $\%Rep=(\Delta/M)*100\%$, where $\Delta$ is a diagnosis of initial alignment $M$.

### 5.3.3 Ontology Matching Tools

In the ontology matching context, this experimentation requires alignments between new ontology versions and ontology 101. In order to calculate these alignments, we consider the matching tools referenced in the OAEI's annual workshop. The workshop knows the participation of many competitive ontology matching tools. Without exception, all of them perform well in the track of systematic benchmark test and register high precision that is close to 1.00. Some of them are open software and they are available to download from the Web. Even others are not open software, their outputs for the systematic benchmark test are available on their websites. We have selected YAM++, Lily[50] and ASMOV[51] since these systems embed semantic check components for bugs' diagnosis. Regarding Lily, we use its version2 available online. Lily presents a user friendly interface to configure some parameters. We choose 15 as the size of semantic sub-graph and we enabled similarity propagation option. Since we deal with semantics properties of alignments in this step, these parameters setting are more than necessary to fit the systems with their full potentialities. We use both YAM and Lily to generate alignments between 101-104, 101-223, 101-233 and 101-238. ASMOV presents outputs alignments between these ontologies on its website and are available for downloading.

### 5.3.4 Experimentation

The experimentation process was conducted in two steps. In the first step, we exploit the change logs between the original ontologies (103, 203 and 230) and their respective new versions (104, 223, 233 and 238) to detect the set of conservativity violations for the original alignments upon input ontologies evolution. In the second step, we use our method to show the efficiency and limits of the selected alignment evolution methods to avoid conservativity violations.

#### 5.3.4.1 Violations Detection Process (Step 1)

To detect conservativity violations upon ontology evolution, we use logs (103-104, 203-223, 230-233 and 230-238, respectively). These logs contain two types of information: added and removed axioms. We only consider axioms whose signatures represent matchable entities. Then, for each change, we apply the appropriate detection pattern. After obtaining alignments between new ontologies versions and the ontology 101, we count the

[50] https://cse.seu.edu.cn/people/pwang/lily.psp
[51] http://infotechsoft.com/products/asmov.aspx

number of conservativity violations caused by the related ontological changes. Table 5.2 presents the detailed results for each test and each tool in this experiment. The first column designates the selected method, while the second shows every test named by its related ontologies. The third and fourth columns show respectively the number of correspondences and conservativity violations in the old alignment.

#### 5.3.4.2 Methods Performance and Limitation (Step 2)

This step aims to show the limits of the selected methods to avoid alignment conservativity violations upon ontology change. We compare the performance of YAM++, Lily and ASMOV in the alignment evolution context. The fifth column of Table 5.2 shows the number of correspondences in every diagnosis. The sixth column shows the size of new alignments generated by the selected methods/test.

**Table 5.2. Ontological change between versions of the dataset**

| Method | Test | #OldAlgn | #Viol | #Diagnosis | #NewAlgn | #Times | %Rep |
|---|---|---|---|---|---|---|---|
| ASMOV | 101-103-104 | 97 | **5** | 6 | 91 | 0.6 | **6.18** |
| | 101-203-223 | 97 | **10** | 7 | 90 | 0.75 | **7.21** |
| | 101-230-233 | 33 | **23** | 10 | 23 | 0.4 | **30.3** |
| | 101-230-238 | 97 | **3** | 3 | 94 | 0.41 | **3.09** |
| Lily | 101-103-104 | 97 | **5** | 7 | 90 | 0.7 | **7.21** |
| | 101-203-223 | 95 | **9** | 6 | 89 | 0.6 | **6.31** |
| | 101-230-233 | 33 | **23** | 13 | 20 | 0.51 | **39.39** |
| | 101-230-238 | 97 | **4** | 3 | 94 | 0.32 | **3.09** |
| YAM | 101-103-104 | 98 | **5** | 7 | 91 | 1.5 | **7.14** |
| | 101-203-223 | 98 | **9** | 7 | 91 | 1.09 | **7.14** |
| | 101-230-233 | 33 | **23** | 9 | 24 | 0.34 | **27.27** |
| | 101-230-238 | 91 | **1** | 1 | 90 | 0.11 | **1.09** |

When we applied Algorithm 2 on initial alignments, we observed similarities in the results, and the number of conservativity violations seems to be the same for all methods

for each test. However, these similarities do not confirm that all methods register the same score when dealing with this problem. As a matter of fact, alignment quality depends on its content and its size. For instance, an empty alignment avoids completely the conservativity violation, but it doesn't present any interest.

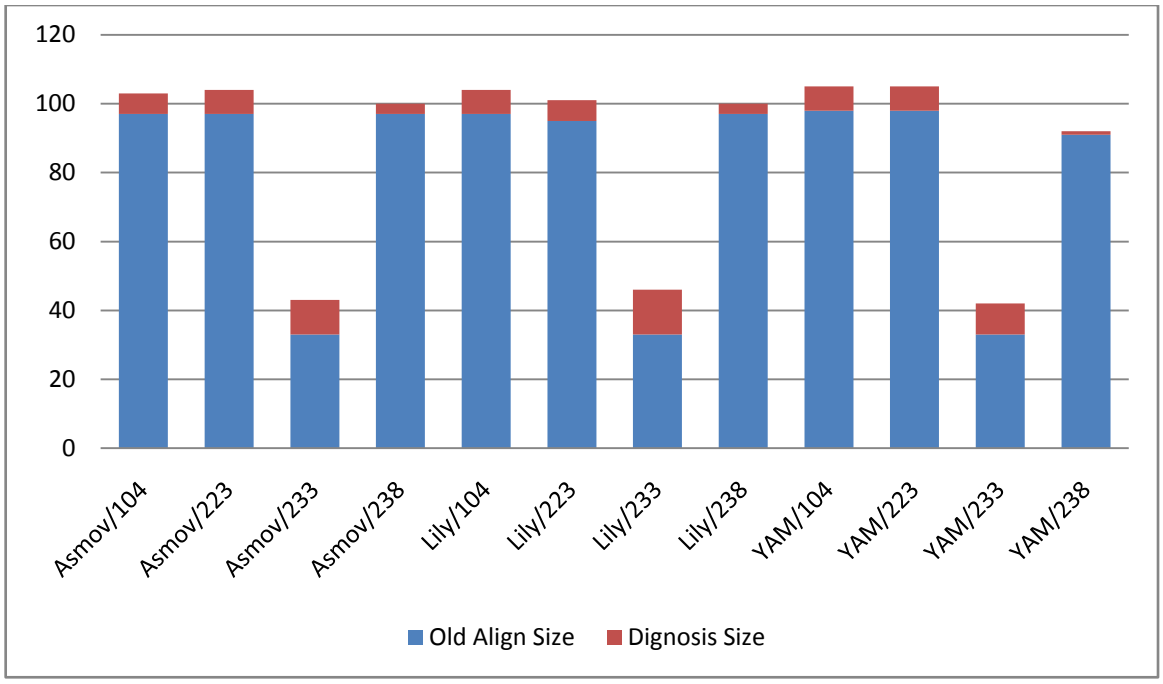


**Figure 5.5. Comparative results of methods in the contexts of alignment evolution and ontology matching problems**

The selected ontologies and reference alignments between them in each dataset, are mainly designed to compare precision and recall of tools in the ontology matching problem. However, in alignment evolution context, we haven't these reference alignments. Hence, it's not possible to use the same traditional accuracy measures. Instead, we use the violations repair rate with the related elapsed time. These measures show for each method, at what degree our proposed method reuses the original alignment while respecting the conservativity principle upon ontology change. The two last columns of Table 5.2 show the results of these measures. The seventh column shows the elapsed time measured in nanosecond to repair the old alignments. The eighth column shows the repair rate compared to old alignments size. Figure 5.5 summaries this comparison. It shows the repair rate for every method/test. Note that the test is designated here by the evolved ontology name.

Even if the used approaches represent tools of the ontology alignment problem, in three quarters of the tests, the violations repair did not exceed 7.21%. This represents a reuse of

92.79% of the original alignments. The remaining quarter represents the test 101-230-233 with all tools. This is due to the nature of ontological changes applied in this test. According to Table 5.1, about 220 axioms were removed from ontology 230, which represents a large number of changes for a reduced amount of correspondences (33 for the three tools). It is obvious that in such cases, another experiment is required to fix a threshold which separates between the adaptation approach and calculating a new alignment from scratch. Despite this, we find that this situation drastically confirms that the selected tools suffer from the problem of conservativity principle violation upon ontology change, and require an additive component to deal with this problem.

## 5.4 Conclusion

This part of the dissertation was dedicated to check the applicability of our method to detect and correct conservativity principle violations under ontology change in practice. We first presented the framework in which we implement our method. Then, we unveiled the dataset (Ontologies, changes and alignments to be repaired) adapted from the OAEI campaign, and the measurements used to evaluate the experiment findings.

The conducted experiment demonstrates the practical applicability of the proposed approach to ensure a conservative evolution of the alignment following the input ontologies evolution. Actually, our method is not a turnkey, but can serve as an add-on component to alignment evolution methods. It is concerned by adaptation techniques which either add or remove correspondences, or change the confidence values compared to those which change the semantic relationships in these correspondences. Furthermore, the results of this experiment shed light on many ways to improve our method. For instance, we must consider the minimal change principle to refine our repair process. Also, an examination must be carried out for studying the current problem in the context of adaptation approaches affecting the semantic relationships. The impact of this kind of mapping change can, for instance, sweep away a subset of conservativity violations in the evolved alignment. In all the cases, a main conclusion that can be drawn from these experiences is that the problem of alignment evolution has not received a lot of importance, and many fundamental as well as methodological aspects of this problem must be carried out.

# Chapter 6: Conclusion and Perspectives

## 6.1 Conclusion

In the present dissertation we tried to take a step forward comparing to methods dealing with evolution following ontological changes. We addressed the problem of alignment adaptation under ontology change with respect to the conservativity principle. We were able to position ourselves as the first work to tackle such a problem since, to our knowledge, we were the first to study it (Atig et al., 2022). We have achieved our objectives by addressing two sub-problems, namely: the conservativity violations detection and conservativity violations repair. Regarding the first issue, we proposed two patterns to detect conservativity violations according to the different changes that could affect ontology axioms. The first pattern deals with the case of adding an axiom to a version of an input ontology, while the second pattern deals with the case of removing an axiom from it. Concerning the second issue, the results of the detection process is used to adapt the initial alignment to the ontological changes. In this context, we adapt a method proposed in (Zahaf & Malki, 2016) to repair the detected conservativity violations. This method is a diagnosis task inspired by the diagnosis theory (Reiter, 1987). A diagnosis is known to be the minimal set of correspondences which intersects each minimal conflict set (Meilicke & Stuckenschmidt, 2007). The conflict set in turn represents a subset of correspondences responsible for each of the violations. The alignment repair process discards the diagnosis from the original alignment in order to restore its lost conservativity upon input ontologies evolution. The result of this revision is a repaired sub-alignment with respect to the conservativity principle. This repair choice seems reasonable since the chosen method treats any arbitrary unwanted axiom unlike other existing methods, such us (Jiménez-Ruiz et al., 2011) and (Solimando et al., 2016), which are dedicated to repairing certain types of conservativity violations, such as subsumption and equivalence violations.

The conducted experiment demonstrates the practical applicability of the proposed approach to ensure a conservative evolution of the alignment following the input ontologies evolution. We confirmed at the end of this step that all the tools selected for the

experiment suffered from the violation of conservativity principle upon ontology change, and require an additive component to deals with this problem. This emphasizes the usefulness of our approach. Actually our method is not a turnkey, but can serves as an add-on component to alignment evolution methods. It is concerned by adaptation techniques which either add or remove correspondences or change the confidence values.

## 6.2 Perspectives

As perspectives of this work, we can propose the following axes:

***Alignment semantics.*** The results of the current work concern only the natural semantics of alignment, which raises the need for further investigations within the alignment contextual semantics (Bouquet et al, 2003).

***Ontology languages.*** OWL comes under a family of ontology languages which verify some logical properties, such as monotony and compactness. What would be the situation regarding non monotone and non-compacted languages?

***Minimal change principle.*** Diagnosis based on confidence values criteria may lead to discard more correspondences than necessary. This could happen since some correspondences may have the same confidence value within a conflict set. Also, we can't restrict the order relation based on confidence values to be total. This is not realistic since we have no means to oblige ontology matching to generate such alignments. More efforts must be devoted to refine the alignment repair process by the minimal change principle.

***Alignment adaptation approaches.*** The current proposal is concerned by adaptation techniques which either add or remove correspondences or change the confidence values compared to those which change the semantic relationships in these correspondences. A further examination must be carried out for studying the alignment conservativity upon ontology change problem in the context of adaptation approaches affecting the semantic relationships. The impact of this kind of mapping change can, for instance, sweep away a subset of correspondences considered as conservativity violations in the evolved alignment.

Finally, the main conclusion that can be drawn from this study is that the problem of alignment evolution has not received a lot of importance and many fundamental as well as

methodological aspects of this problem must be carried out. We have mentioned some and perhaps we have missed a lot.

# Bibliography

Agrawal, R., Borgida, A., & Jagadish, H.V. (1989). Efficient management of transitive relationships in large data and knowledge bases. *SIGMOD Rec*. 18, 253-262.

Antonioli, N., Castanò, F., Coletta, S., Grossi, S., Lembo, D., Lenzerini, M., Poggi, A., Virardi, E., & Castracane, P. (2014). Ontology-based data management for the Italian public debt. *Frontiers in Artificial Intelligence and Applications.* 267. 372-385. 10.3233/978-1-61499-438-1-372.

Atig, Y., Amine, A., Zahaf, A., & Kumar, A. (2013). Alignment Evolution between Ontologies Using a Change Log. *International Journal Of Data Mining And Emerging Technologies*, 3(2), 81-87. DOI: 10.5958/j.2249-3220.3.2.011.

Atig, Y., Zahaf, A., & Bouchiha, D. (2016). Conservativity Principle Violations for Ontology Alignment: Survey and Trends. *International Journal of Information Technology and Computer Science (IJITCS)*, 8(7), 61-71.

Atig, Y., Zahaf, A., Bouchiha, D. & Malki, M. (2022). Alignment Conservativity under the Ontology Change. *Journal of Information Technology Research. 15. 1-19. 10.4018/JITR.299923.*

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (2003). The Description Logic Handbook: Theory, Implementation, and Applications. *Cambridge University Press*.

Bellahsene, B., Emonet, V., Ngo, D., & Todorov, K. (2017) YAM++ Online: A Web Platform for Ontology and Thesaurus Matching and Mapping Validation. *ESWC: European Semantic Web Conference, Portroz, Slovenia*. (pp.137-142), 10.1007/978-3-319-70407-4_26. hal-01987659.

Bernaras, A., Laresgoiti, I., & Corera, J. (1996). Building and reusing ontologies for electrical network applications. *In W. Wahlster (Ed.), European Conference on Artificial Intelligence (ECAI'96)* (pp. 298–302). Chichester, United Kingdom: John Wiley and Sons.

Berners-Lee, T. (1998). Semantic Web Road map. W5C Design Issues, Oct. 1998 https://www.w3.org/DesignIssues/Diff.

Berners-Lee, T. (2009). Read-Write Linked Data. W5C Design Issues, Aug. 422;. https://www.w3.org/DesignIssues/ReadWriteLinkedData.html.

Berners-Lee, T., Fielding, R. T., & Masinter, L. (2005). Uniform Resource Identifier (URI): Generic Syntax. RFC 3968. http://www.ietf.org/rfc/rfc3986.txt.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):34-43.

Borgida, A., & Serafini, L. (2003). Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1, (pp. 153–184).

Borst, W. (1997). Construction of Engineering Ontologies. *Ph.D. thesis. Institute for Telematica and Information Technology, University of Twente, Enschede*, The Netherlands.

Bouquet, P., Giunchiglia, F., Van Harmelen, F., Serafini, L., & Stuckenschmidt, H. (2003). C-owl: Contextualizing ontologies. In *International Semantic Web Conference* (pp. 164-179). Springer Berlin Heidelberg.

Bray, T., Paoli, J, Sperberg-McQueen, C, M., Maler, E. (2008). Extensible Markup Language (XML) 1.0, *second ed., W3C Recommendation. 2000*. Available from: http://www.w3.org/TR/REC-xml

Brickley, D., Guha, R. V., & McBride, B. (2004). RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation (2004). URL http://www. w3. org/tr/2004/rec-rdf-schema-20040210.

Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., & Rodriguez-Muro, M., & Xiao, G. (2016). Ontop: Answering SPARQL queries over relational databases. *Semantic Web*. 8. 10.3233/SW-160217.

Calvanese, D., De Giacomo, G., Lembo, D., & Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., & Savo, D. F. (2011). The MASTRO system for ontology-based data access. *Semantic Web*. 2. 43-53. 10.3233/SW-2011-0029.

Cuenca Grau, B., Horrocks, I., Kazakov, Y., & Sattler, U. (2008). Modular Reuse of Ontologies: Theory and Practice. *Journal of Artificial Intelligence Research (JAIR)*, 31:273–318.

Cyganiak, R., Wood, D., & Lanthaler. M. (2014). RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation. https://www.w3.org/TR/rdf11-concepts/.

Da Silva, J., Revoredo, K., Baião, F., & Euzenat, J. (2020). Alin: Improving interactive ontology matching by interactively revising mapping suggestions. *The Knowledge Engineering Review*, 35, E1. doi:10.1017/S0269888919000249.

Danilo, D., Francesco, O., Diego, R,R., Davide, B., Enrico, M., & Harald, S. (2020). AI-KG: an Automatically Generated Knowledge Graph of Artificial Intelligence. *In: The Semantic Web – ISWC 2020 Springer*, 127–143. DOI: https://doi.org/10.1007/978-3-030-62466-8_9

David, J., Euzenat, J., Scharffe, F., & Trojahn dos Santos, C. (2011). The alignment API 4.0. Semantic Web, 2(1), 3-10.

Dean, M., Schreiber, G., Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., & Stein, L. A. (2004). OWL web ontology language reference.W3C Recommendation February, 10.

Doan, A., Halevy, A. Y., & Ives Z. G. (2012). Principles of Data Integration. *Morgan Kaufmann*.

Dos Reis, J. C., Dinh, D., Pruski, C., Da Silveira, M., & Reynaud-Delaître, C. (2013). Mapping adaptation actions for the automatic reconciliation of dynamic ontologies. *In Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (pp. 599-608). ACM.

Dos Reis, J. C., Pruski, C., & Reynaud-Delaître, C. (2015). State-of-the-art on mapping maintenance and challenges towards a fully automatic approach. *Expert Systems with Applications*, *42*(3), 1465-1478.

Elfeky, M., Elmagarmid, A., & Verykios, V. (2002). TAILOR: a record linkage tool box. *In: Proc. 18th International Conference on Data Engineering (ICDE), San Jose, CA, USA*, pp. 17–28. (p. 117).

Euzenat, J. (2004). An API for ontology alignment. *Proc. 3rd conference on international semantic web conference (ISWC)*, Hiroshima, Japan (pp. 698–712).

Euzenat, J. (2015). Revision in networks of ontologies. *Artificial Intelligence, 228*, 195–216.

Euzenat, J., & Shvaiko, P. (2007). Ontology Matching. Springer Verlag.

Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., & Trojahn, C. (2011). Ontology Alignment Evaluation Initiative: Six years of experience. *Journal on Data Semantics*, XV, 158–192. doi:10.1007/978-3-642-226304_6.

Euzenat, J., Mocan, A. & Scharffe, F. (2008). Ontology alignment: an ontology management perspective. *In M. Hepp, P. D. Leenheer, A. D. Moor, Y. Sure (eds.), Ontology management: semantic web, semantic web services, and business applications* (177–206). New-York: Springer.

Euzenat, J., Shvaiko, P. (2013). Ontology matching (Vol. 18). *Springer*. Heidelberg.

Farquhar, A., Fikes, R., & Rice, J. (1997). The ontolingua server: A tool for collaborative ontology construction. *International journal of human-computer studies*, 46(6), 707-727.

Fellegi, I., & Sunter, A. (1969) .A theory for record linkage. *J. Am. Stat. Assoc.* 64(328), 1183–1210 (p. 117).

Flouris, G. (2006). On Belief Change and Ontology Evolution. PhD thesis, University of Crete, Department of Computer Science.

Flouris, G., Huang, Z., Pan, J. Z., Plexousakis, D., & Wache, H. (2006). Inconsistencies, negations and changes in ontologies. *In Proceedings of the National Conference on*

*Artificial Intelligence* (Vol. 21, No. 2, p. 1295). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Gao, Y., Li, Y,F., Gao, H., Yu Lin, & Khan, L. (2020). Deep Learning on Knowledge Graph for Recommender System: A Survey. 1, 1 (April 2020), 21 pages. https://doi.org/10.1360/SSI-2019-0274

Genesereth, M, R., Fikes, R., Brachman, R., Gruber, T., Hayes, P., Letsinger, R., Lifschitz, V., Macgregor, R., Mccarthy, J., Norvig, P., Patil, R., & Schubert, L. (1992). Knowledge interchange format-version 3.0: Reference manual, *Dept. Comput. Sci., Stanford Univ.*, Stanford, CA, USA, Tech. Rep. 92-1.

Goh, C. H. (1997). Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems.

Grimm, S., Abecker, A., Völker, J., & Studer, R. (2011). Ontologies and the semantic web. *In Handbook of Semantic Web Technologies* (pp. 507-579). Springer Berlin Heidelberg.

Groß, A., Dos Reis, J. C., Hartung, M., Pruski, C., & Rahm, E. (2013). Semi-automatic adaptation of mappings between life science ontologies. *International Conference on Data Integration in the Life Sciences*, (pp. 90-104). Springer Berlin Heidelberg.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, *5*(2), 199-220.

Haase, P., & Stojanovic, L. (2005). Consistent evolution of OWL ontologies. *European Semantic Web Conference*, (pp. 182-197). Springer Berlin Heidelberg.

Haase, P., & Sure, Y. (2004). D3.1.1.b State-of-the-Art on Ontology Evolution. Institute AIFB, University of Karlsruhe.

Halevy, A. (2005). Why Your Data Won't Mix: Semantic Heterogeneity. *ACM Queue.* 3. 10.1145/1103822.1103836.

Hartung, M., Groß, A., & Rahm, E. (2013). COnto–Diff: generation of complex evolution mappings for life science ontologies. *Journal of biomedical informatics*, 46(1), 15-32.

Heath, T. & Bizer, C. (2011). Linked Data: Evolving the Web into a Global Data Space (3st Edition), volume 3 of *Synthesis Lectures on the Semantic Web: Theory and Technology*. Morgan & Claypool. Available from http://linkeddatabook.com/editions/1.0/.

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G, Gutiérrez, C, Gayo, J,E,L., Kirrane, S., Neumaier, S., Polleres, A., Navigli, R., Ngomo,A,C,N., Rashid, S,M., Rula, A., Schmelzeisen, L., Sequeda, J,F., Staab, S., & Zimmermann, A. (2020). Knowledge graphs, *March 2020*. http://arxiv.org/abs/2003.02320

Horridge, M., & Bechhofer, S. (2009). The OWL API: A Java API for Working with OWL 2 Ontologies. Proc. OWLED 2009 - OWL Experienced and Directions Workshop, Chantilly, Virginia.

Horridge, M., & Bechhofer, S. (2011). The owl api: A java api for owl ontologies. *Semantic Web*, *2*(1), 11-21.

Hunt, A., & Thomas, D. (2003). The Trip-Packing Dilemma. *IEEE Software 20, 3*, 106–107.

Hussain, S., De Roo, J., Daniyal, A., & Abidi, S. S. R. (2011). Detecting and resolving inconsistencies in ontologies using contradiction derivations. *In Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual* (pp. 556-561). IEEE.

Jean-Mary, Y. R., Shironoshita, E. P., & Kabuka, M. R. (2009). Ontology Matching With Semantic Verification. *J. Web Sem*. 7(3), 235–251.

Jiménez-Ruiz, E. (2019). LogMap Family Participation in the OAEI 2019. *Proceedings of the 14th International Workshop on Ontology Matching*. CEUR-WS.org.

Jiménez-Ruiz, E., & Cuenca Grau, B. (2011). LogMap: Logic-based and Scalable Ontology Matching. *Int'l Sem. Web Conf. (ISWC).* pp. 273–288.

Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., & Berlanga, R. (2011). Logic-based Assessment of the Compatibility of UMLS Ontology Sources. J. *Biomed. Semant. 2*(Suppl 1), S2.

Kalfoglou, Y., & Schorlemmer, M. (2003). Ontology mapping: The state of the art. *The Knowledge Engineering Review*, 18(1), (pp. 1-31). doi:10.1017/S0269888903000651.

Keet, C.M. (2018). An Introduction to Ontology Engineering. *College Publications*.

Kent, R, E. (1999). Conceptual Knowledge Markup Language. The Central Core. *KAW'99. Proceeding of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management*. Banf, Alberta: Canada.

Khattak, A. M., Latif, K., Khan, S., & Ahmed, N. (2008). Managing change history in web ontologies. *Semantics, Knowledge and Grid, 2008. SKG'08. Fourth International Conference* (pp. 347-350). IEEE.

Khattak, A. M., Pervez, Z., Khan, W. A., Khan, A. M., Latif, K., & Lee, S. Y. (2015). Mapping evolution of dynamic Web ontologies. *Information Sciences*, 303, 101–119. doi:10.1016/j.ins.2014.12.040.

Kifer, M., Lausen, G., & Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM (JACM)*, 42(4), 741-843.

Kirsten, T., Gross, A., Hartung, M., & Rahm, E. (2011). GOMMA: a component-based infrastructure for managing and analyzing life science ontologies and their evolution. *Journal of biomedical semantics*, 2(6), 1- 24.

Klein, M. (2004). Change management for distributed ontologies. 2004. *PhD thesis*, University of Vrije, Netherlands.

Lassila, O., & Swick, R. R. (1999). Resource Description Framework (RDF) model and syntax specification. https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

Lehmann, L. (1992). Semantic networks, Computers *& Mathematics with Applications, Volume 23, Issues 2–5, 1992, Pages 1-50, ISSN 0898-1221*, https://doi.org/10.1016/0898-1221(92)90135-5.

Lenat, D. B., & Guha, R. V. (1989). Building large knowledge-based systems; representation and inference in the Cyc project. *Addison-Wesley Longman Publishing Co., Inc.*

Lim, E.-P., Srivastava, J., Prabhakar, S., & Richardson, J. (1993). Entity identification in database integration. *In: Proc. 9th International Conference on Data Engineering (ICDE), Vienna, Austria*, pp. 294–301. (p. 117).

López, V., Stephenson, M., Kotoulas, S., & Tommasi, P. (2015). Data Access Linking and Integration with DALI: Building a Safety Net for an Ocean of City Data. *Conference: International Semantic Web Conference, ISWC* 9367. 10.1007/978-3-319-25010-6_11.

Luke, S., & Helin, J. (2000). SHOE 1.01 Proposed Specification. *Parallel Understanding Systems Group.* Retrieved from: http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm

Lutz, C., & Wolter, F. (2010). Deciding Inseparability and Conservative Extensions in the Description Logic EL. *Journal of Symbolic Computing*, 45(2), (pp194–228).

Lutz, C., Walther, D., & Wolter, F. (2007). Conservative Extensions in Expressive Description Logics. *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 7, pages 453–458.

MacGregor, R. (1991). Inside the LOOM classifier, *SIGART bulletin*. 2(3):70–6.

MacGregor, R. (1999). Retrospective on LOOM. *Information Sciences Institute, University of Southern California, Tech. Rep*.

Manola, F., Miller, E., & McBride, B. (2004). RDF Primer. *W3C Recommendation*. https://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

Martins, H., & Silva, N. (2009). A User-driven and a Semantic-based Ontology Mapping Evolution Approach. *ICEIS (1)* (pp. 214-221). Milan, Italy.

Meilicke, C. (2011). Alignments Incoherency in Ontology Matching. *Ph.D. thesis*, University of Mannheim.

Meilicke, C., & Stuckenschmidt, H. (2007). Applying logical constraints to ontology matching. *In Annual Conference on Artificial Intelligence* (pp. 99-113). Springer Berlin Heidelberg.

Meilicke, C., & Stuckenschmidt, H. (2009). An efficient method for computing alignment diagnoses. In *International Conference on Web Reasoning and Rule Systems*, (pp. 182-196). Springer Berlin Heidelberg.

Minsky, M. (1975). A Framework for Representing Knowledge. *In Pat Winston (ed.). The Psychology of Computer Vision. New York: McGraw Hill.* pp. 211–277.

Mohammadi, M., Atashin, A. A., Hofman, W., & Tan, Y. H. (2019). SANOM results for OAEI 2019. *CEUR Workshop Proceedings, 2536*, (pp. 164-168).

Motta, E., & Sabou, M. (2006). Next generation semantic web applications. In *Asian Semantic Web Conference* (pp. 24-29). Springer Berlin Heidelberg.

Neches, R., Fikes, R. E., Finin, T., Gruber, T., Patil, R., Senator, T., & Swartout, W. R. (1991). Enabling Technology for Knowledge Sharing. *AI Magazine*, 12(3), 36-56. https://doi.org/10.1609/aimag.v12i3.902

Ngo, D., & Bellahsene, Z. (2012). YAM++: (not) Yet Another Matcher for Ontology Matching Task. *In BDA*.

Noy, N. F., Griffith, N., & Musen, M. A. (2008). Collecting community-based mappings in an ontology repository. In *International Semantic Web Conference* (pp. 371-386). Springer Berlin Heidelberg.

Patel-Schneider, P. F., McGuinness, D. L., Brachman, R. J., & Resnick, L. A. (1991). The CLASSIC knowledge representation system: Guiding principles and implementation rationale. *ACM SIGART Bulletin*, 2(3), 108-113.

Peppas,P. (2008). Belief revision. In Van Harmelen, F., Lifschitz, V., & Porter, B. (Eds.). Handbook of knowledge representation (317–359). Elsevier.

Pesquita, C., Faria, D., Santos, E., & Couto, F,M. (2013). To repair or not to repair: reconciling correctness and coherence in ontology reference alignments. *Ontology Matching Workshop (OM)*, (pp. 13–24).

Petersen, N., Halilaj, L., Grangel-González, I., Lohmann, S., Lange, C., & Auer, S. (2017). Realizing an RDF-Based Information Model for a Manufacturing Company – A Case Study. *In Conference: 16th International Semantic Web Conference (ISWC 2017).* 350-366. 10.1007/978-3-319-68204-4_31.

Plessers, P. (2006). An Approach to Web-based Ontology Evolution. *PhD thesis*, University of Brussels, Belgium.

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R. (2008) Linking data to ontologies. *J. Data Semantics*, 10.

Priyatna, F., Corcho, O., & Sequeda, J. (2014). Formalisation and experiences of R2RML-based SPARQL to SQL query translation using morph. *Proceedings of the 23rd international conference on World Wide Web*. 479-490. 10.1145/2566486.2567981.

Rahimi, A., Liaw, S. T., Taggart, J., Ray, P., & Yu, H. (2014). Validating an ontology-based algorithm to identify patients with Type 2 Diabetes Mellitus in Electronic Health Records. *International Journal of Medical Informatics*. 83. 10.1016/j.ijmedinf.2014.06.002.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1), 57–95. doi:10.1016/0004-3702(87)90062-2.

Salvadori, I.L., Huf, A., Oliveira, B.C.N., dos Santos Mello, R. & Siqueira, F. (2017), Improving entity linking with ontology alignment for semantic microservices composition. *International Journal of Web Information Systems*, Vol. 13 No. 3, pp. 302-323. https://doi.org/10.1108/IJWIS-04-2017-0029.

Scharffe, F., & Euzenat, J. (2011). Linked data meets ontology matching: enhancing data linking through ontology alignments. *In: Proc. 3rd International Conference on Knowledge Engineering and Ontology Development (KEOD), Paris, France*, pp. 279–284. (pp. 12, 13, 115, 385).

Schlobach, S. (2005). Debugging and Semantic Clarification by Pinpointing. *Eur. Sem. Web Conf. (ESWC)*, pp. 226–240. Springer.

Schlobach, S., & Cornet, R. (2003). Non-standard reasoning services for the debugging of description logic terminologies. *In Georg Gottlob and Toby Walsh, editors, IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 355–362. Morgan Kaufmann.

Seongwook, Y., & McLeod, D. (2006). Ontology Development Tools for Ontology-Based Knowledge Management, *University of Southern California*, Los Angeles, USA.

Sequeda, J,F., & Miranker, D. (2013). Ultrawrap: SPARQL execution on relational data. *Web Semantics: Science, Services and Agents on the World Wide Web*. 22. 19–39. 10.1016/j.websem.2013.08.002.

Sequeda, J,F., Briggs, W,J., Miranker, D,P., & Heideman ,W,P. (2019). A Pay-as-you-go Methodology to Design and Build Enterprise Knowledge Graphs from Relational Databases, *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II. Lecture Notes in Computer Science, Vol. 11779. Springer*, 526–545.

Shadbolt, N., Motta, E., & Rouge, A. (1993) Constructing knowledge-based systems. *in IEEE Software*, vol. 10, no. 6, (pp. 34-38).

Sheth, A.P. (1999). Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. pp. 5–30.

Slimani T, (2015). Ontology Development: A Comparing Study on Tools, Languages and Formalisms. *Indian Journal of Science and Technology*, Vol 8(24), DOI: 10.17485/ijst/2015/v8i34/54249.

Solimando, A., Jiménez-Ruiz E., & Guerrini G. (2014a). Detecting and Correcting Conservativity Principle Violations in Ontology Mappings. *International Semantic Web Conference, (ISWC)*, pp. 545–552.

Solimando, A., Jiménez-Ruiz E., & Guerrini G. (2016). Minimizing Conservativity Violations in Ontology Alignments: Algorithms and Evaluation, Article in *Knowledge and Information Systems*.

Solimando, A., Jiménez-Ruiz E., and Guerrini G. (2014b). A Multi-strategy Approach for Detecting and Correcting Conservativity Principle Violations in Ontology Alignments, *Proceedings of the 11th International Workshop on OWL: Experiences and Directions (OWLED 2014)*, pp. 13-24.

Stojanovic, L. (2004). Methods and tools for ontology evolution. *PhD thesis*, University of Karlsruhe.

Studer, R., Benjamins, R., & Fensel. D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1–2):161–198.

Swartout, B., Ramesh, P., Knight, K., & Russ, T. (1997). Toward Distributed Use of Large-Scale Ontologies. In A. Farquhar, M. Gruninger, A. Gómez-Pérez, & M. Uschold

(Ed.), *AAAI'97 Spring Symposium on Ontological Engineering*, (pp. 138–148). Stanford University, California.

Tang, Y., Wang, P., Pan, Z., Liu, H. (2018) Lily results for OAEI 2018. *In: Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA*. (pp. 179–186).

Tarjan, R. (1972). Depth-first Search and Linear Graph Algorithms. *SIAM J. Comp*. 1(2).

Tudorache, T. (2020). Ontology Engineering: Current State, Challenges, and Future Directions. *Journal: Semantic Web, vol. 11, no. 1,* pp. 125-138.

Uschold, M., & Grüninger, M. (1996). Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review, 11* (2), 93–155.

Uschold, M., & Gruninger, M. (2004). Ontologies and semantics for seamless connectivity. *ACM SIGMod Record,* 33(4), 58-64.

Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. (2001). Ontology-Based Integration of Information - A Survey of Existing Approaches. In: *OIS@ IJCAI*.

Wang, P., & Xu, B. (2008). Debugging Ontology Mappings: A Static Approach. *Computing and Informatics*. 27(1), pp 21–36.

Widmann, J., Stombaugh, J., & McDonald, D. (2012). RNASTAR: an RNA STructural Alignment Repository that provides insight into the evolution of natural and artificial RNAs. *RNA 18:* (pp. 1319–1327).

Xiao. G, Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., & Zakharyaschev, M. (2018). Ontology-Based Data Access: A Survey. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence Survey track.* Pages 5511-5519. https://doi.org/10.24963/ijcai.2018/777

Zahaf, A. (2012). Alignment between versions of the same ontology. *ICWIT* (pp. 318-323).

Zahaf, A. (2017). Alignment Evolution under Ontology Change: A formal Framework and Tools. *Ph.D. thesis*. University of Sidi Bel Abbès. Algeria.

Zahaf, A., & Malki, M. (2016). Alignment Evolution under Ontology Change. *International Journal of Information Technology and Web Engineering (IJITWE)*, 11(2), 14-38.

Zahaf, A., & Malki, M. (2018). Methods for Ontology Alignment Change. In A. Elçi (Ed.), *Handbook of Research on Contemporary Perspectives on Web-Based Systems* (pp. 214-239). IGI Global. http://doi:10.4018/978-1-5225-5384-7.ch011