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 **ZAHAF AHMED**

Spécialité : *INFORMATIQUE*
Option : *INFORMATIQUE*

Intitulée

Alignment Evolution under Ontology Change: A formal Framework and Tools

Soutenue le 08 Juin 2017
Devant le jury composé de :

Président :

Mme. SLAMA	**Zohra**	**M.C.A**	**UDL-SBA**	**Président**

Examinateurs :

Mr. BENSLIMANE	**Sidi Mohamed**	**Prof.**	**ESI-SBA**	**Examinateur**
Mr. AMAR BENSABER	**Djamel**	**M.C.A**	**ESI-SBA**	**Examinateur**
Mr. TOUMOUH	**Adil**	**M.C.A**	**UDL-SBA**	**Examinateur**
Mr. BOUKLI	**Hacene**	**M.C.A**	**UDL-SBA**	**Examinateur**

Directeur de thèse :

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

Année universitaire : 2016-2017

بسم الله الرحمن الرحيم

الحمد لله الذي وفقني لإتمام هذا العمل و الصلاة و السلام على رسوله الكريم محمد بن عبد الله الذي لولاه ما حمدت ربي.

اللهم اجعل هذا العمل في سبيلك.

إلى كل عالم أو متعلم أو محب لهما.

إلى روح والدتي الطاهرة، اللهم اجعل هذا العمل رحمة عليها.

إلى والدي.

إلى زوجتي وأولادي ملاك ، محمد ، وسيم و لين. أرجو أن يسامحوني على انشغالي عنهم طوال مدة هذا العمل.

Acknowledgement

First of all, I would like to express my sincere thanks to Prof. Dr. MALKI Mimoun, my supervisor, for his encouragement and support during my research.

I would like to thank Dr. SLAMA Zohra, (UDL-SBA) who was willing to serve on my dissertation committee as president. Furthermore, I thank the professors Prof. Dr. BENSLIMANE Sidi Mohamed (ESI-SBA), Dr. AMAR BENSABER Djamel (ESI-SBA), Dr. BOUKLI HACENE Sofiane (UDL-SBA), and Dr. TOUMOUH Adil (UDL-SBA), who served on the examination committee.

I would also like to thank all the researchers I met and with whom I discussed at least one of the points concerning this thesis. I especially want to thank the Professor Atilla Elçi for his generosity, his advice and his warm welcome in his department at Aksaray University. I also thank the Professor Michel Simonet for his warm welcome in his laboratory. I thank his wife the Professor Ana and her daughters for the hospitality in their home in Grenoble. I also thank the Professor Ueno for his warm welcome in his laboratory at the National Institute of Informatics in Tokyo. I also thank all the researchers I met during the days of the summer school on the semantic web and ontologies in Madrid. I especially thank the Professor Jérome Euzenat (INRIA - Grenoble) with whom I conducted a fruitful discussion that has clarified very much to me the light to find my way while trying to understand the problem of alignment evolution.

Last but not least, I thank my colleagues Dr. Aissa Fellah and Dr. Bouchiha Djelloul with whom I was in constant mutual discussion on problems related to our doctoral theses.

Abstract. Alignment of ontologies is the backbone of semantic interoperability. It facilitates the import of data from an ontology to another, translating queries between them, or merging ontologies in a global one. However, these services cannot be guaranteed throughout the life cycle of the ontology. The problem is that the evolution of aligned ontologies may affect and make obsolete the alignment. Contributions of this dissertation touch the literature review and the methodology knowledge sides of the alignment evolution problem. At the methodology knowledge side, the dissertation proposes a formal framework that consists of a number of phases, each having a specific purpose. The framework facilitates ontology change identification for maintainers. On the light of base revision theory, the framework presents a set of generic operators for evolving alignments from a consistent state to another consistent state with a minimal of change. The framework adapts the Hitting set algorithm of diagnosis theory to concretize these operators. Besides, the framework is extended with a global method which is an orchestration of a set of operations each of which is designed to take care of one aspect of the alignment change process. Finally, the framework allows to maintainers reviewing the change before implementation. In what concerns the literature review side, the dissertation mentions the importance of the problem and recommends the separation of its study from the study of the ontology evolution problem. Besides, the dissertation suggests classifying the alignment evolution approaches in two categories. Approaches of the former are corrective since they check and resolve inconsistences after change while approaches of the latter are adaptive and perfective since they only adapt the alignment according to detected changes in ontologies. Moreover, the dissertation demonstrates the advantage of the proposed approach relatively to others. The results show that neither ontology matching nor alignment debugging methods fit well for the alignment evolution problem.

Keywords: Ontological Change, Alignment Evolution, Alignment Revision, Belief Base Revision, Diagnosis theory.

Résumé. L'alignement des ontologies est l'épine dorsale de l'interopérabilité sémantique. Il facilite l'importation de données d'une ontologie à une autre, traduisant des requêtes entre elles, ou fusionner les ontologies. Cependant, ces services ne peuvent pas être garantis tout au long du cycle de vie de l'ontologie. Le problème est que l'évolution des ontologies alignées peut affecter et rendre obsolète l'alignement. Les contributions de cette dissertation touchent le côté de l'analyse de la littérature ainsi que le côté méthodologique du problème de l'évolution de l'alignement. Sur le côté méthodologique, la thèse propose un cadre formel qui consiste en un certain nombre de phases, chacune ayant un but spécifique. Le cadre du travail facilite l'identification des changements ontologiques pour les ingénieurs de maintenance de l'alignement. À la lumière de la théorie de la révision des bases de croyances, il présente un ensemble d'opérateurs génériques pour l'évolution de l'alignement d'un état consistant à un autre état consistant avec un minimum de changement. Il adapte l'algorithme Hitting Set de la théorie du diagnostic pour concrétiser ces opérateurs. En outre, le cadre est étendu avec une méthode globale qui est une orchestration d'un ensemble d'opérations dont chacune est conçue pour prendre soin d'un aspect du processus de changement de l'alignement. Enfin, il permet aux ingénieurs de maintenance d'examiner le changement avant de sa mise en œuvre. En ce qui concerne l'analyse de la littérature, la dissertation mentionne l'importance du problème et recommande la séparation de son étude du problème de l'évolution de l'ontologie. En outre, la thèse propose de classer les approches d'évolution de l'alignement en deux catégories. Les approches de la première sont correctives puisqu'elles vérifient et résolvent l'inconsistance de l'alignement. Alors que les approches de la deuxième sont adaptatives et perfectives, puisqu'elles n'ont qu'adapter l'alignement en fonction des changements détectés dans les ontologies. De plus, la thèse démontre l'avantage de l'approche proposée par rapport aux autres. Les résultats montrent que ni les outils de matching ni les méthodes de débogage de l'alignement ne conviennent bien au problème de l'évolution de l'alignement.

Mots Clés: Changement d'ontologie, Evolution de l'alignement, Révision de l'alignement, Révision des bases de croyances, Théorie des diagnostics.

ملخص:تعتبر مطابقة الأنطولوجيات العمود الفقري للعمل المشترك الدلالي. فهو يسهل استيراد البيانات من أنطولوجيا إلى آخرة، ترجمة الاسئلة بينهما أو دمج الأنطولوجيات في واحدة. إن هذه الخدمات لا يمكن أن تكون مضمونة في جميع مراحل دورة حياة الأنطولوجيا.ذلك أن تطور الأنطولوجيات قد يؤثر على التطابق بينهمم ويجعله غير صالح للإستعمال.

المساهمات المعرفية لهذه الأطروحة تلمس مراجعة الجانب الأدبي وكذلك الجانب المنهجي الخاص بمشكلة تطور مطابقة الأنطولوجيات. فيمايخص الجانب المنهجي ، تقترح الأطروحة إطارا رياضيا يتكون من عدة مراحل، ولكل منها غرض محدد. يسهل الإطار تحديد التغييرات التي تطرأ على الأنطولوجيا. على ضوء نظرية مراجعة الإعتقادات ،يعرض الإطار عمليتين عامتين لتطور مطابقة الأنطولوجيات بشكل متناسق مع الضمان لمبدأ الحد الأدنى من التغيير. بتكييف خوارزمية ضرب المجموعة من نظرية التشخيص يجسد الإطار عدد كبير من العمليات. الى جانب ذلك، تم تزويد الإطار بطريقة شاملة على شكل تناسق مجموعة من العمليات كل منها مصمم للتكفل بجانب واحد من عملية تطور مطابقة الأنطولوجيات. يسمح الإطارأيضا بمراجعة التغييرالمقترح قبل التنفيذ.

في ما يتعلق بمراجعة الجانب الأدبي، تشدد الأطروحة على أهمية المشكلة وتوصي بضرورة فصل دراستها عن مشكلة تطور الأنطولوجيات. الى جانب ذلك، تقترح الأطروحة تصنيف مقاربات مشكلة تطور مطابقة الأنطولوجيات الى فئتين. تهدف الفئة الأولى الى تصحيح التطوربتحديد وحل التناقضات بعد التغيير. في حين أن الفئة الثانية تسعى الى تكييف المطابقة وفق التغييرالحاصل في الأنطولوجيات دون مراعات التناقضات بشكل واضح. وعلاوة على ذلك، تظهرالأطروحة أفضلية المقاربة المطروحة على بعض المقاربات الأخرى. وأظهرت النتائج أنه لا وسائل حساب مطابقة الأنطولوجيات ولا طرق تصحيحها تناسب بشكل جيد مشكلة تطور مطابقة الأنطولوجيات.

الكلمات المفتاحية : تغير الأنطولوجيا، تطوير تطابيق الأنطولوجيات ، مراجعة تطابيق الأنطولوجيات ، مراجعة قواعد الإعتقادات، نظرية التشخيص

Contents

List of figures

List of tables

Chapitre 1 Introduction

1.1 Semantic Interoperability

Ontologies play an important role in many computer applications where it is necessary to overcome the problem of heterogeneity and diversity in semantics. They define a formal semantics for the information enabling the semantic interoperability of information sources (Fensel, 2001). With the emergence of the semantic web, ontologies have proliferated and are accessible to a wide audience. This leads to the appearance of several, but overlapping ontologies for the same domain and each source of information is free to choose the most appropriate ontology to its needs. Consequently, the information interoperability problem turns to a problem of ontology interoperability. Ontology interoperability can be achieved by ontology matching tools (Euzenat & Shvaiko, 2013) which aim at finding semantic correspondences between related entities of different ontologies. These correspondences express semantic relations between entities of different ontologies. The set of these correspondences constitutes an alignment between ontologies. An alignment is used to import data from an ontology to another, translating queries between them or merging ontologies in a global one (Euzenat et al., 2008).

Early semantic web applications such as AquaLog (Lopez et al., 2005) and Magpie (Dzbor et al., 2003) use ontologies and alignments between them in the design time. Thanks to ontology repositories such as Swoogle (Ding et al., 2004), Watson (d'Aquin et al., 2007), OntoSelect (Buitelaar et al., 2004), the DAML ontology library[1] and Schema.org[2] that store, index, organize and share ontologies, a new generation of applications (Motta & Sabou, 2006) can find and use dynamically the appropriate ontologies in the run-time. For instance, PowerAqua (Lopez et al., 2006) (the successor of AquaLog) is a cross-domain question

[1] http://www.daml.org/ontologies/
[2] http://schema.org/

answering system. It locates thanks to Watson[3], online semantics documents that match user's queries. Besides ontologies, some repositories such as Bioportal[4] and Alignment server[5] consider alignments as a first class object, enhancing the dynamic interoperability of ontologies. They, store, index, organize and share alignments. These infrastructures allow applications to seek and use on the fly the appropriate alignments.

1.2 Problem Statement

Ontologies are continuously in evolution so that they reflect our gradual understanding of reality (Hepp, 2007). The evolution reflects changes in the application domain or in the business strategy and incorporating additional functionality according to changes in the users' needs (Stojanovic, 2004). The dynamic aspect of ontologies can affect and make obsolete the alignment between them. This is a special case of the known alignment evolution problem (Euzenat & Shvaiko, 2013; Dos Reis et al., 2015; Groß et al., 2013; Dos Reis et al., 2013; Martins &Silva, 2009). We call such a case as the alignment evolution under ontology change problem. This problem can be refined to include others sub-problems.

Problem 1(ontology change identification): In open and distributed environments such as the semantic web where ontologies and alignments are submitted to different authorities, the ontology change is often available in an unreadable machine format or not delivered at all. Even the ontology evolution approaches (Stojanovic, 2004; Plessers, 2006; Klein, 2004) deliver evolution logs that store the implemented change, maintainers of alignments may not share the same interpretation for the same change and they prefer to create their own set of change which might be different from the delivered set of changes. Maintainers want to identify and make explicit the ontology change in order to understand what happen and correctly update their alignments.

Problem 2(alignment consistency): As ontologies evolve from a consistent state to another, an alignment evolution should follow this change by transition to

[3] http://watson.kmi.open.ac.uk/WatsonWUI/
[4] http://bioportal.bioontology.org
[5] http://alignapi.gforge.inria.fr/aserv.html

a new consistent state too. The alignment consistency is expressed as a set of constraints qualified as hard since their violation makes obsolete the alignment and useless. We highlight the complexity of this problem via scenarios clarified with simple and meaningful examples.

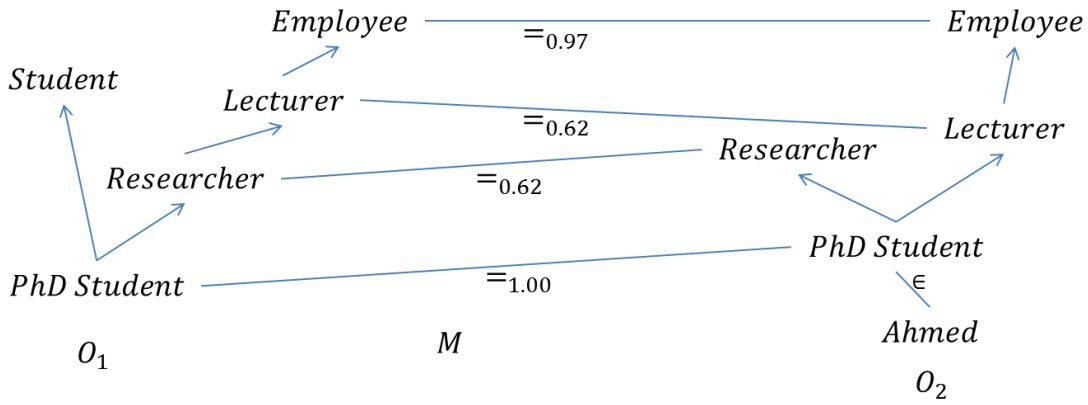

Figure 1: An example of an alignment between two educational domain ontologies.

Scenario 1: alignment correspondences refer only to entities that belong to the aligned ontologies. The deletion of these ontological entities breaks the structure of the concerned correspondences. An alignment which has such correspondences is structurally inconsistent. An alignment should preserve its structure after the ontology change. We call such constraint, the structure preservation constraint. Example 1 describes a scenario where the ontology change can violate this constraint.

Example 1: Considering the alignment M of Figure 1. We use Description Logic like syntax to describe both ontologies. Also, we use the index number in ontologies notation as name space to designate entities. The ontologies O_1 and O_2 and the alignment M are also described as follows.

$$O_1 = \begin{Bmatrix} PhD\ Student \sqsubseteq Researcher, \\ PhD\ Student \sqsubseteq Student, \\ Researcher \sqsubseteq Lecturer, \\ Lecturer \sqsubseteq Employee \end{Bmatrix}, O_2 = \begin{Bmatrix} PhD\ Student \sqsubseteq Researcher, \\ Lecturer \sqsubseteq Employee, \\ PhD\ Student \sqsubseteq Lecturer, \\ PhD\ Student\ (Ahmed) \end{Bmatrix}$$

$$M = \begin{Bmatrix} 1: PhD\ Student =_{1.00} 2: PhD\ Student, \\ 1: Researcher =_{0.62} 2: Researcher, \\ 1: Lecturer =_{0.62} 2: Lecturer, \\ 1: Employee =_{0.97} 2: Employee \end{Bmatrix}$$

Assuming the designer of ontology O_2 decides to remove the concept Employee. The new version is:

$$O_2' = \begin{Bmatrix} PhD\ Student \sqsubseteq Researcher, \\ PhD\ Student \sqsubseteq Lecturer, \\ Phd\ Student\ (Ahmed) \end{Bmatrix}$$

However, the alignment M contains the correspondence $1{:}\,Employee =_{0.97} 2{:}\,Employee$ while O_2' don't contain the concept $2{:}\,Employee$. We say the alignment M is structurally inconsistent.

Scenario 2: ontologies are logical theories. Even, ontologies ensure their logical consistencies after the change; they can't preserve this consistency when they are used jointly with alignment. For preserving the logical consistency of ontologies, we should prevent the alignment from generating these inconsistencies as logical consequences. We call such constraint, the logical consistency preservation. The following scenario highlights this problem:

Example 2: Following example 1, the designer sets the concepts Researcher and Lecturer as disjoints. So, he revises O_2' to add the axiom $Researcher \perp Lecturer$. Now, the new version is

$$O_2'' = \begin{Bmatrix} PhD\ Student \sqsubseteq Researcher, \\ PhD\ Student \sqsubseteq Lecturer, \\ Lecturer \perp Researcher, \\ Phd\ Student\ (Ahmed) \end{Bmatrix}$$

He remarks that O_2'' is inconsistent. He justifies this inconsistency by the following sequence: from $O_2'' \models$ $Phd\ Student\ (Ahmed), PhD\ Student \sqsubseteq Lecturer, PhD\ Student \sqsubseteq Researcher$, he can derive $O_2'' \models Lecturer\ (Ahmed), Researcher\ (Ahmed)$ but $O_2'' \models Lecturer \perp Researcher$. Hence, he decides to revise O_2'' such that PhD Students will no longer be lecturers. To do that, he removes the axiom $2{:}PhD\ Student \sqsubseteq Lecturer$ and he obtain the new version

$$O_3 = \begin{Bmatrix} PhD\ Student \sqsubseteq Researcher, \\ Lecturer \perp Researcher, \\ Phd\ Student\ (Ahmed) \end{Bmatrix}.$$

When he tries to use the alignment M he concludes that $M \models Lecturer(Ahmed), Researcher(Ahmed)$. *Therefore, the alignment M is inconsistent. These two axioms are alignment consequences of M. Indeed, from 2:Phd Student(Ahmed), 2:Phd Student* $\sqsubseteq$ *Researcher, 2:Researcher* $\equiv$ *1:Researcher, 1:Researcher* $\sqsubseteq$ *Lecturer, 1:Lecturer* $\equiv$ *2: Lecturer, he derives 2:Lecturer(Ahmed). From 2:Phd Student(Ahmed), 2: Phd Student* $\equiv$ *1: Phd Student, 1: Phd Student* $\sqsubseteq$ *Researcher, 1:Researcher* $\equiv$ *2:Researcher, he derives 2: Researcher(Ahmed).*

Scenario 3: ontologies are the pillar of the semantic web; alignments maintainers may have not the permission to modify the changed ontologies in order to establish the consistency of alignments. In other words, alignments maintainers should accept the ontological change and modifying alignments is the only possible way to establish the new consistency. Accepting the change may not be respected if some removed knowledge still entailed by alignments. In both cases, alignments should follow the ontology change by preserving it. We call such constraint, the ontological change preservation.

Example 3: In example 2, the designer accepts the ontological change by considering the new version O_3 *instead of the previous versions* O_2' *and* O_2''. *In this case, it is easy to verify that* $M \models Lecturer \perp Researcher$. *One justification of inconsistency of the alignment M is even the axiom 2:PhD Student* $\sqsubseteq$ *Lecturer has been removed, it stills entailed by M as the following logical consequences demonstrate. From* $M \models$ *2:PhD Student* $\sqsubseteq$ *Researcher, 2:Researcher* $\equiv$ *1:Researcher, 1:Researcher* $\sqsubseteq$ *Lecturer, 1:Lecturer* $\equiv$ *2:Lecturer we can derive* $M \models$ *2: PhD Student* $\sqsubseteq$ *Lecturer. We say the alignment M preserve the change in the former while violates the change preservation constraint in the latter.*

Problem 3 (minimality of change): Many solutions can satisfy the consistency constraints when we evolve the alignment. One of them is the empty alignment where we discard all its correspondences. The empty alignment doesn't make any sense from a practical point of view and we need to compute the new alignment from scratch. An ideal solution is to change only the relevant correspondences that

cause problems. We call such constraint, the constraint of minimal change. In contrast with the consistency constraints which are qualified as hard we qualify the minimal change as a soft constraint. Since the violation of this constraint don't hamper the use of alignments.

Problem 4 (User involvement): alignment evolution is a knowledge intensive task which can't be fulfilled without the involvement of users. The system proposes the change and maintainers are invited to review it before implementation. Maintainers may validate the change, recover the unnecessary changes, adapt, track, or cancel the change. Hence, the system should facilitate the interaction with users and enhance the interoperability with others tools.

1.3 Position and contributions

Recently, some approaches (Groß et al., 2013; Dos Reis et al., 2013; Martins and Silva, 2009) have emerged to deal with the problem of alignment evolution under ontology change. The main challenge of these approaches is how to adapt the alignment following an ontology change. Influenced by the underlying representation of ontology, logical properties of alignment are neglected. Considering ontologies as logical theories allows a recent approach (Euzenat, 2015) to define a formal and general framework for alignment revision mirroring the AGM[6] model (Alchourrón et al., 1985) of belief revision theory. In this framework, ontologies are closed sets under the logical consequence of the underlying semantics of alignment. However, ontologies and hence alignments are encoded in knowledge bases making applications only holds a subset of domain knowledge as explicit and using reasoning services to derive implicit ones. This practical representation of ontologies and alignments leads us to consider a different approach based on base revision theory (Dalal, 1988; Hansson, 1994; Hansson, 2006) to deal with this problem.

Contributions and accomplishments that are the result of this dissertation touch the literature review and the methodology knowledge sides of the alignment

[6] AGM model is the most influential work in belief revision theory (see chapter 2 for more details).

evolution problem. In what concerns the literature review side, our contribution is two-fold.

- We have reviewed the main ontology evolution frameworks. Guiding by the fixed requirements of the alignment evolution problem, we have concluded that these frameworks should be adapted in order to embed the alignment evolution problem. Moreover, we have recommended that the alignment evolution problem should be separated from the ontology evolution problem since alignment depending artifacts may create confusion with depending artifacts of ontologies.
- Inspired by the classification of the software evolution and maintenance approaches in software engineering, our second contribution is the classification of the alignment evolution approaches in three classes: adaptive, corrective, and perfective maintenance. After review, we observed all approaches fall in two categories. The approaches of the former are corrective since they check and resolve inconsistences after change. The main challenge for these approaches is how to ensure a consistency alignment with a minimal of change. While the approaches of the latter are adaptive and perfective since they don't consider explicitly the alignment consistency and they only adapt the alignment according to the detected changes in ontologies. Consequently, no guaranties are given to ensure the alignment consistent even they claim it was their primary purpose.

At the methodology knowledge side, the dissertation presents a new approach for the alignment evolution under ontology change problem. The approach proposes a general framework that consists of a change process with fourth phases: a phase for ontology change identification, a phase for the semantics of change, a phase for change validation, and a phase for change implementation. Number of methods, each having a specific purpose, are designed for concretizing the change process.

- The framework proposes a new method for the ontology change identification which is published first in (Zahaf, 2012) and reproduced later in (Zahaf and Malki, 2016b). This method compares between versions of the same ontology and delivers the change as the changed vocabulary in one

hand and the changed axiomatic meaning of this vocabulary in the other hand. This format of change which constitutes an ontology of change helps for understanding and sharing the change.

- To resolve the logical consistency and the change preservation consistency the framework adapts the kernel framework (Hansson, 1994; Hansson, 2006) of belief base revision theory to design a variety of operators. These operators base their actions on notions of alignment kernel and incision function. The framework adapts the Hitting set algorithm (Reiter, 1987) of diagnosis theory to compute the alignment kernel as well as the corresponding incision functions. This part of the framework is published in (Zahaf and Malki, 2016a). For the purpose of satisfying the structural consistency, the framework only suggests the removing of the concerned correspondences.

- We are satisfied by proposing a weak form of the principle of minimal change. The designed operators for the alignment consistency resolution satisfy core-retainment postulate which means only correspondences that participate somehow in the inconsistency implication need to be changed. Sometimes, not all these correspondences should be changed but only a subset of them. This is why we need the user involvement to achieve the operation of alignment change.

- Besides inconsistency checking, our system of alignment evolution counts on notions of the kernel and change log to facilitate the interaction with users and enhance its interoperability with tiers. The former plays the role of inconsistency explanations while the latter allows change tracking and change sharing with depending applications.

- Besides the different proposed operators of change, the framework is extended with a global method which is an orchestration of a set of operations each of which is designed to take care of one aspect of the alignment change process. The method is published in (Zahaf and Malki, 2016b).

- Mainly, some approaches from the adaptive and perfective category rely on ontology matching techniques for evolving alignments. The dissertation

demonstrates the advantage of our approach relatively to these approaches. Results which are published in (Zahaf and Malki, 2016b) show that neither ontology matching nor alignment debugging methods fit well for the alignment evolution problem.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 constitutes the background of the thesis. Its content is two-folds: belief revision theory and diagnosis theory. The belief revision theory presents change theory in an elegant logical formalism and the diagnosis theory complement theory with practices to fulfill real-world applications. In chapter 3, we explore the state of the art of works done to resolve the problem of alignment evolution. We take the analysis of these approaches in accordance with the aforementioned requirements of the alignment evolution problem. Our framework is the subject of chapter 4. First, we present our proposal for the alignment change process in which, we identify and formalize the constraints and requirements that the alignment change should satisfy. Then, we introduce the models of ontologies and alignments considered in this framework. We connect the alignment revision under ontology change problem and base revision theory. Meanwhile, we justify our choice to follow base revision theory. Always, on the light of this theory, we design a set of operations that satisfy these constraints and requirements. Chapter 5 represents the computational aspect of our framework. We give a plethora of algorithms and methods with different complexities varying from exponential time to polynomial time for concretizing the designed operators. In Chapter 6, we discuss a prototype implementation of our system of alignment evolution. Then we conduct an experimental process to demonstrate the advantage of our approach relatively to others. Finally, we summarize our thesis and we give an outline about future works in chapter 7.

Chapter 2. Background knowledge

2.1 Introduction

Belief change is one of the several names that are being used to denote a matured research field of how an agent rationally changes his beliefs. Some of the others are theory change, theory revision, belief change, and belief revision. Belief change research relies on nice and precise logical formalism, but it lacks the implementation of realistic revision methods. Diagnosis theory is another field of research with the objective to restore systems consistency after deficiencies. The research field of diagnosis theory has powerful tools to prune computational complexity to the problem of diagnosis, allowing them to deal with real-world situations. However, many applications lack clear formalizations. Cross-field applications have shown the relation of belief revision theory with diagnosis theory practices. In this chapter, we present briefly both theories in two separate sections. The AGM model is the most influential work in belief revision research field. Section 2.2 describes this model and its extension of belief base revision as well. Section 2.3 presents the diagnosis theory according to the approach of diagnosis from the first principle.

2.2 Belief change

2.2.1 History

Belief change is a philosophical discipline beginning in 1970's to discuss the requirements of rational belief change. Two milestones can be highlighted (Hansson, 1999). The former is a series of studies conducted by Levi and Harper (as cited in Hansson, 1999). Problems posed by Levi have since been the main concerns of this research area. Alchourrón and Makinson (as cited in Hansson, 1999) had previously cooperated in studies of changes in legal codes. Gärdenfors early work (as cited in Hansson, 1999) was concerned with the connections

between belief change and conditional sentences (if-sentence). Facilitated by these works, the trio Carlos Alchourrón, Petre Gärdenfors, and David Makinson combined forces and developed the AGM model which is a general and versatile formal framework for studies of belief change (Alchourrón et al., 1985). This model gave the naissance of the second milestone of research on belief change. The model has been extended beyond the studies of changes in legal codes to meet new fields such as database updates and knowledge engineering. This practical mutation was the direct impact that gave birth to the new model of base revision theory. Database updating was largely influenced by the development of artificial intelligence. Fagin et al. (1983) introduced the notion of 'database priorities' for updating databases with integrity constraints. The truth maintenance systems developed by Doyle (1979) was also important in this development. The AGM and base revision models rely on a nice and a precise logical formalism to define a variety of change operators each of which is characterized by a set of postulates to constraint the performed change.

2.2.2 Belief representations

Beliefs are represented by sentences in some formal language. Hansson (2006) claims "Sentences do not capture all aspects of belief, but they are the best general-purpose representation that is presently available. The beliefs held by an agent are represented by a set of such belief-representing sentences". The formal language defines a relation C_n called logical consequence on this set of beliefs. For any set K of beliefs, $C_n(K)$ is the set of beliefs that follow logically from K. We say a belief α is a logical consequence of K if and only if $\alpha \in C_n(K)$. We also use $K \vDash \alpha$ to denote this. In what follows, K will denote a belief set. $K \nvDash \alpha$ for $\alpha \notin (K)$. $C_n(\phi)$ is the set of tautologies.

Two types of sets are used to represent the beliefs. The former, just called belief set[7] is closed under logical consequence (Alchourrón et al., 1985). Formally,

Definition 2.1 (Belief sets): a belief set K is a set closed under logical consequence if and only if $K = C_n(K)$.

[7] "In logic, logically closed sets are called theories. In formal epistemology, they are also called corpora, knowledge sets, or (more commonly) belief sets" (Hansson, 2006).

Every sentence that follows logically from this set is already in the set. If the language is sufficiently rich, a belief set might become very large to a point of containing beliefs that the agent has never thought of. Furthermore, if the language is infinite, then so does the belief set (Hansson, 1999). This usually means dealing with infinite beliefs set which cannot be incorporated easily into a computational framework (Peppas, 2008). Human minds and actual computers can only hold a finite subset of beliefs that may (roughly) correspond to the explicit beliefs (Hansson, 1999). Such sets are called belief bases which consist of sets not necessarily closed under logical consequence. The formal definition is as follows:

Definition 2.2 (Belief bases): Any set B of sentences is a belief base. Let K be a belief set. Then a set B of sentences is a belief base for K if and only if $K = C_n(B)$.

According to Hansson (1999), "belief bases are not required by definition to be finite, but in all realistic applications they will be so". Nevertheless, some beliefs can only be derived from others one. The elements of the base represent the basic beliefs that are held independently of any others beliefs. Those elements of its logical closure that are not elements of the belief base itself are called the derived beliefs (Hansson, 2006). In contrast to a belief set, a sentence α is belief of a base if and only if it is a consequence of that belief base, $\alpha \in C_n(B)$. In set-theoretical language:

α is a belief if and only if $\alpha \in C_n(B)$.

α is a belief base if and only if $\alpha \in B$.

α is a (merely) derived belief if and only if $\alpha \in C_n(B)/B$.

2.2.3 Belief set change

AGM (Alchourrón et al., 1985) is the most influential model in belief set revision research (Fermé & Hansson, 2011). In the AGM model, beliefs are represented as sentences of a formal language L governed by a Tarskian logic. For any set K of sentences, a Tarskian consequence operation on a given language is a function C_n from sets of sentences to sets of sentences. It satisfies the following three conditions:

Inclusion: $K \subseteq C_n(K)$

Monotony: If $K \subseteq K'$, then $C_n(K) \subseteq C_n(K')$

Iteration: $C_n(K) = C_n(C_n(K))$

AGM assumptions

In the AGM theory, a little is assumed about this language and its logic. The language should be closed under all Boolean connectives. The language contains the usual truth-functional connectives: the negation ($\neg$), the conjunction ($\wedge$), the disjunction ($\vee$), the implication ($\rightarrow$), and the equivalence ($\leftrightarrow$) while the logic is assumed to be:

Supraclassicality: if α can be derived from K by classical truth-functional logic, then $\alpha \in C_n(K)$.

Deduction : for all $\alpha \in L$ and $K \subseteq L$, if $\alpha \in C_n(K \cup \{\beta\})$ then $\beta \longrightarrow \alpha \in C_n(K)$.

Compactness : for all $\alpha \in L$ and $K \subseteq L$, if $\alpha \in C_n(K)$ then there is some subset $K' \subseteq K$ such that $\alpha \in C_n(K')$.

Types of change

Three types of belief change are defined: expansion, revision, and contraction. ***Expansion***: if no inconsistency occurs when adding a belief to the previous set, the expansion consists in a set-theoretical adding of the new beliefs. Formally, the expansion of K by a sentence α is the operation that just adds α and removes nothing, is denoted $K + \alpha$ and defined as follows: $K + \alpha = C_n(K \cup \{\alpha\})$.

While expansion can be defined in a unique way, there exists a class of operators for belief revision, as well as for contraction. Every class is characterized by a set of postulates and a set of constructors that should satisfy these postulates.

Revision: a revision change should incorporate new beliefs while ensuring consistency of the new set of beliefs. The belief revision is modeled as a function $*$ mapping a theory K and a sentence α to a new theory $K * \alpha$. In order to capture the notion of rational belief revision, some constraints are imposed on belief revision operators. The principle of minimal change is an intuition guide in the formulation of these constraints. According to this principle, a rational agent

should change his beliefs as little as possible to consistently receive the new information (Peppas, 2008). Gärdenfors(1992) formulated a set of eight postulates, known as the AGM postulates for belief revision.

The outcome of change should be a theory. This postulate is called the closure. Formally,

$K * \alpha = C_n(K * \alpha)$ (Closure).

The new information α should successfully be included in the new belief set. This postulate is called the success. Agents trust enormously on the reliability of the new information in a way that this new information outweighs any previous contradictory beliefs, whatever those beliefs may be.

$\alpha \in K * \alpha$ (Success).

The new belief set $K * \alpha$ will be a subset of the whole of K, the new information α, all whatever follows from the logical closure of K and α, and nothing more. This is formulated in the postulate of inclusion:

$K * \alpha \subseteq K + \alpha$ (Inclusion)

Whenever the new information α does not contradict the initial belief set K, there is no reason to remove any of the original beliefs at all;

If $\neg \alpha \notin K$ then $K + \alpha \subseteq K * \alpha$ (Vacuity).

Essentially, the vacuity postulate expresses the notion of minimal change in the limiting case where the new information is consistent with the initial beliefs.

Belief revision aims for consistency at any cost unless the new information in itself is inconsistent. In which case, because of the success postulate, the revision can't do anything about the consistency.

If α is consistent then $K * \alpha$ is also consistent (consistency).

The syntax of the new information has no effect on the revision process; all that matters is the proposition it represents. This postulate says the revision change is irrelevant of the syntax. Hence, logically equivalent sentences α and β change a theory K in the same way. Formally,

If $\vDash \alpha \leftrightarrow \beta$ then $K * \alpha = K * \beta$ (Extensionality).

The six above postulates are called the basic postulates. Gärdenfors (1992) suggested that revision should also satisfy two further supplementary postulates. The idea is that, if $K * \alpha$ is a revision of K and $K * \alpha$ is to be changed by a further sentence β, such a change should be made by expansions of $K * \alpha$ whenever possible. More generally, the minimal change of K to include both α and β, that is, $K * \alpha \wedge \beta$, ought to be the same as the expansion of $K * \alpha$ by β, so long as β does not contradict the beliefs in $K * \alpha$. For technical reasons the precise formulation is split into two postulates:

$K * (\alpha \wedge \beta) \subseteq (K * \alpha) + \beta$ (*Superexpansion*).

If $\neg\beta \notin K * \alpha$ then $(K * \alpha) + \beta \subseteq K * (\alpha \wedge \beta)$ (*Subexpansion*).

Contraction: Contraction is choosing what to believe. It is the operation of removing a specified belief from the belief set. Like belief revision, the belief contraction is formally defined as a function $-$ mapping a theory K and a sentence α to a new theory $K - \alpha$. Once again a set of eight postulates was proposed, motivated by the principle of minimal change, to constraint $-$ in a way that captures the essence of rational belief contraction. These postulates, known as the AGM postulates for belief contraction, are the following:

When a belief set K is contracted by a sentence α, the outcome should be logically closed.

$K - \alpha = C_n(K - \alpha)$ (Closure).

Inclusion ensures no new beliefs should be added to the contracted set:

$K - \alpha \subseteq K$ (Inclusion)

If the sentence to be contracted is not included in the original belief set, then contraction by that sentence involves no change at all. Such contractions should leave the original set unchanged.

If $\alpha \notin K$ then $K - \alpha = K$ (Vacuity)

The postulate success says that the retracted belief should not be believed after contraction unless it is a tautology. Contraction should be successful, i.e., $K - \alpha$ should not imply α (or not contain α, which is the same thing if Closure is satisfied). However, it would be too much to require that $\alpha \notin C_n(K - \alpha)$ for all

sentences α, since it cannot hold if α is a tautology. The success postulate has to be conditional on α not being logically true.

If $\nvDash \alpha$ then $\alpha \notin K - \alpha$ (Success)

The contraction with syntactically different but logically equivalent sentences should be the same. This is so called the extensionality postulate. Extensionality tells us that contraction is syntax independent.

If $\vDash \alpha \leftrightarrow \beta$ then $K - \alpha = K - \beta$ (Extensionality)

Belief contraction should be minimal in the sense of keeping as many beliefs as possible in the original beliefs set. One way for guaranteeing this principle is to require that everything can be recovered exactly to the same state before the contraction when expanding the contracted set again by the same belief α. This is so called the recovery postulate which enables the change undo.

If $\alpha \in K$, then $K \subseteq (K - \alpha) + \alpha$ (Recovery)

The six above cited postulates are called the basic postulates. Here again, two further postulates relate the individual contractions by two sentences α and β, to the contraction by their conjunction $\alpha \wedge \beta$. To contract K by $\alpha \wedge \beta$, we need to give up either α or β or both. All beliefs that survive the contraction by α as well as the contraction by β should not be affected by the contraction with their conjunction $\alpha \wedge \beta$. This postulate is formulated as follows:

$(K - \alpha) \cap (K - \beta) \subseteq K - (\alpha \wedge \beta)$ (Conjunctive overlap).

Finally, assume that $\alpha \notin K - (\alpha \wedge \beta)$. Since $K - \alpha$ is the minimal change of K to remove α, it follows that $K - (\alpha \wedge \beta)$ can not be larger than $K - \alpha$. The last postulate, in fact, makes it smaller or equal to it; in symbols.

If $\alpha \notin K - (\alpha \wedge \beta)$ then $K - (\alpha \wedge \beta) \subseteq K - \alpha$ (Conjunctive inclusion).

Constructors

Constructors for revision can be obtained from contraction constructors by applying Levi identity. We can also get constructors for contraction from revision constructors by applying Harper identity. Belief revision literature presents several constructions for contraction that satisfies the AGM postulates. See

(Gärdenfors, 1992) for more details. In this section, we present one of these constructions called the partial meet contraction which is relevant to the content of this thesis. We prefer to present Levi and Harper identities first.

Levi and Harper Identities

The contraction and revision operators were distinctively characterized by sets of postulates. The postulates for contraction are independent of postulates for revisions and vice versa. Here we present two procedures to get revision operators from contractions operators and vice versa.

A revision of a belief set can be seen as a composition of a contraction and an expansion. More precisely, In order to construct the revision $K * \alpha$, one first contracts K with respect to $\neg\alpha$ and then expands $K - \neg\alpha$ by α. Formally, we have the following definition which is called Levi identity:

$$K * \alpha = (K - \neg\alpha) + \alpha$$

Conversely, a contraction can be defined in term of a revision. The idea is that a sentence δ is accepted in the contraction $K - \alpha$ if and only if δ is accepted both in K and in $K * \neg\alpha$. Formally, this amounts to the following definition which has been called Harper identity:

$$K - \alpha = K \cap K * \neg\alpha$$

Partial Meet Contraction

The outcome of contracting K by α should be a subset of K that does not imply α, but from which no elements of K have been unnecessarily removed. By applying the principle of minimal change, the contracted belief set $K - \alpha$ should be as large a subset of K as it can be without implying α. In general, there exist more than one such maximal subset of K. The set of such maximal subsets of K that do not imply α is called remainder set. More precisely,

Definition 2.3 (Remainder set): Let K be a set of sentences and α a sentence. The remainder set of K by α noted $K \perp \alpha$ (K less α) is the set such that $A \in K \perp \alpha$ if and only if:

$$\begin{cases} A \sqsubseteq K \text{ (it is a subset of K)} \\ A \nvDash \alpha \text{ (that don't imply } \alpha) \\ \forall A', A \subset A', A' \vDash \alpha \text{ (and it is maximal)} \end{cases}$$

If the principle of minimal change is strictly applied, then the outcome of contracting K by α should be an element of $K \perp \alpha$. An operation that satisfies this property is called a choice contraction (Alchourrón et al., 1985).

Since $K \perp \alpha$ typically has many elements, we need a selection function to choose among them. A selection function selects elements of $K \perp \alpha$ unless $K \perp \alpha$ is empty. Formally,

Definition 2.4 (Selection function): a selection function γ for K is a function that for all sentences α:

$$\begin{cases} \emptyset \neq \gamma(K \perp \alpha) \sqsubseteq (K \perp \alpha), \text{if } K \perp \alpha \neq \emptyset, \\ \gamma(K \perp \alpha) = \{K\} \text{ otherwise} \end{cases}$$

If the selection function selects exactly one element from $K \perp \alpha$, we can formulate the choice contraction as follows:

$$K - \alpha = \gamma(K \perp \alpha).$$

The choice contraction doesn't allow the believer to contract cautiously. Full meet contraction is an alternative contraction to choice. It forces the believer to be cautious in all situations (Hansson, 1999). It is formulated as follows:

$$K - \alpha = \bigcap(K \perp \alpha).$$

The partial meet contraction is the intermediate solution between the extreme caution of the full meet contraction and the extreme incautiousness of the choice contraction (Hansson, 1999). It uses a selection function as in choice contraction, but it allows it to choose several elements of $K \perp \alpha$. The partial meet contraction is then the intersection of selected elements of $K \perp \alpha$. Formally,

Definition 2.5 (Partial meet Contraction): let K be a set of sentences and α a sentence and γ is a selection function, the partial meet contraction of K by α is the operator defined as $K -_{\gamma} \alpha = \bigcap \gamma(K \perp \alpha)$.

Alchourrón et al (1985) show that every partial meet contraction satisfies the postulates of AGM model and every contraction that satisfies AGM postulates

should be a partial meet contraction. This result is known in the literature as the representation theorem for the partial meet contraction:

Theorem 2.1 (Representation theorem): an operator − is a partial meet contraction for a belief set K if and only if it satisfies the postulates of closure, inclusion, vacuity, success, extensionality, and recovery.

A variant of the partial meet contraction that satisfies all AGM postulates is the transitively relational partial meet contraction. Such an operator defines a transitive reflexive ordering relation ($\leq$) on the remainder set $K \perp \alpha$. The idea is to constrain the selection function to select the top elements of $K \perp \alpha$.

2.2.4 Belief base change

In AGM theory, all beliefs are equal vis-à-vis the change. However, changes are performed only on basic beliefs not on derived beliefs in base revision theory. The underlying intuition is that a derived belief is not worth retaining if loses the support that it had in basic beliefs. Then it will be automatically discarded (Hansson, 2006).

Nebel (as cited in Peppas, 2008) distinguishes between approaches that aim to take into account the difference between explicit and derived beliefs on one hand and approaches that aim to provide a computational model for theory revision on the other. The former (Hansson, 1999; Hansson & Wassermann, 2002) gives rise to belief base revision operations, whereas the latter (Nebel, 1994) defines belief base revision schemes. The main difference between these approaches is their outcomes. The output of a belief base revision operation is again a belief base. However the output of a belief base revision scheme is a theory.

We consider in this thesis belief base revision operations. Like AGM model, the belief base revision accepts the same types of change: expansion, revision, and contraction. Unlike AGM model, every constructor is characterized by a set of postulates that is different from the set of postulates of another constructor. Constructors in belief base revision are not equivalents. In what follows, we introduce some operators as well as the set of postulates that characterize every operator.

Base expansion

The expansion of a belief base B by a sentence α is the operation that just adds α and removes nothing, is denoted $B + \alpha$ and defined as follows:

$$B + \alpha = B \cup \{\alpha\}.$$

Base contraction

Given a belief base B and a particular belief α, the objective of contraction is to compute a subset of B that fails to imply α. Hansson (1999) introduced two operators for the base contraction: the Partial meet base contraction and the kernel contraction.

Partial meet base revision

The operator of partial meet base contraction is similar to the partial meet contraction for belief sets. The only difference is to apply the operator on belief base B instead of belief set K. For a selection function γ, the partial meet base contraction of B by α, is:

$$B -_{\gamma} \alpha = \bigcap \gamma(B \perp \alpha)$$

Hansson (1999) characterizes the partial meet base contraction by the following postulates:

Theorem 2.2 (Partial meet base Contraction Representation): the operator $-$ is a partial meet base contraction for a belief base B if and only if it satisfies the following postulates:

[success] if $\nvDash \alpha$ then $B - \alpha \nvDash \alpha$

[Inclusion] $B - \alpha \subseteq B$

[uniformity] if it holds for all $B' \subseteq B$ that $B' \vDash \alpha$ if and only if $B' \vDash \beta$, then $B - \alpha = B - \beta$

[relevance] if $\beta \in B$ and $\beta \notin B - \alpha$, then there is a subset B' of B such that, $B - \alpha \subseteq B' \subseteq B$ and $B' \nvDash \alpha$ but $B' \sqcup \{\beta\} \vDash \alpha$

As the contraction for a belief set, the partial meet contraction satisfies success and inclusion but it needs to satisfy two new postulates: uniformity and relevance.

The uniformity postulate requires that if every subset that implies some belief α implies also another belief β, then the contraction by α and β should be the same. Uniformity is stronger than extensionality, i.e. it implies extensionality, but it is not implied by it. The postulate recovery can't hold in general in the theory of base revision and is replaced by the relevance postulate. Relevance means only beliefs that are responsible for implying the contracted belief should be discarded.

Kernel Contraction

The kernel contraction is a particular operation of contraction (Hansson, 1994). It consists of finding the set of minimal subset of B that imply α. This set is called the kernel of B by α and denoted by $B \parallel \alpha$. An element of the kernel $B \parallel \alpha$ is called α-kernel. Formally,

Definition 2.6 (Kernel): the kernel of B by α is the set of B' such that:

$$\begin{cases} B' \sqsubseteq B \text{ (it is a subset of B)} \\ B' \vDash \alpha \text{ (that imply } \alpha) \\ \forall B'' \sqsubseteq B', B'' \nvDash \alpha \text{ (and it is minimal)} \end{cases}$$

Then, the kernel contraction uses a function to discard from B at least one element from each α-kernel. The function is called an incision function.

Definition 2.7 (Incision function): an incision function σ for B is a function that for all α:

$$\begin{cases} \sigma(B \parallel \alpha) \sqsubseteq \bigsqcup(B \parallel \alpha) \\ \text{if } \emptyset \neq X \in B \parallel \alpha, \text{then } X \cap \sigma(B \parallel \alpha) \neq \emptyset \end{cases}$$

Definition 2.8 (Kernel Contraction): let B a belief base, α a belief and σ an incision function, the kernel contraction of B by α is the operator defined as $B -_{\sigma} \alpha = B \setminus \sigma(B \parallel \alpha)$

The kernel contraction has proved to satisfy the following postulates (Hansson, 1994): success, inclusion, core-retainment, and uniformity. The following representation theorem summarizes these postulates for every kernel contraction operator.

Theorem 2.3 (Kernel Contraction Representation): the operator $-$ is a kernel contraction for a belief base B if and only if it satisfies the following postulates:

[success] if $\not\vdash \alpha$ then $B - \alpha \not\vdash \alpha$

[Inclusion] $B - \alpha \subseteq B$

[Core − retainment] if $\beta \in B$ and $\beta \notin B - \alpha$, then there is a subset B' of B such that, $B' \not\vdash \alpha$ but $B' \sqcup \{\beta\} \vdash \alpha$

[uniformity] if it holds for all $B' \subseteq B$ that $B' \vdash \alpha$ if and only if $B' \vdash \beta$, then $B - \alpha = B - \beta$

Core-retainment is a weak version of the relevance postulate: Instead of requiring B' to be interposed between B and $B - \alpha$, we are satisfied by requiring it to be a subset of B. While relevance requires that excluded sentence β that in some way contributes to the fact B, but not $B - \alpha$, implies α, core-retainment requires that it contributes to the fact B implies α. The elements of B that do not contribute at all in making B imply α are called the α-core of B (Hansson, 1999).

Base revision

Just like the corresponding operators for belief sets, revision operators for belief bases can be constructed from two sub-operations: an expansion by α and a contraction by $\neg\alpha$ (Hansson, 2006). According to Levi identity ($B * \alpha = (B - \neg\alpha) + \alpha$), the contractive suboperation should take place first. Alternatively, the two sub-operations may take place in reverse order, $B * \alpha = (B + \alpha) - \neg\alpha$. This latter possibility does not exist for belief sets. If $K \cup \{\alpha\}$ is inconsistent, then $K + \alpha$ is always the same (namely identical to the whole language) independently of the identity of K and of α, so that all distinctions are lost. For belief bases, this limitation is not present, and thus there are two distinct ways to base revision on contraction and expansion:

Internal revision: $B * \alpha = (B - \neg\alpha) + \alpha$

External revision: $B * \alpha = (B + \alpha) - \neg\alpha$

Intuitively, the external revision accepts an intermediate inconsistent state in which both α and $\neg\alpha$ are believed, however the internal revision has an intermediate state in which neither α nor $\neg\alpha$ is believed. The two operators differ in their logical properties (Hansson, 2006).

Base consolidation

Consolidation is an operation that makes consistent an inconsistent belief base (Hansson, 1997). Unfortunately, this operation of inconsistent belief bases does not have a plausible counterpart for inconsistent belief sets. The reason is that there is only one inconsistent belief set and all distinctions are lost (Hansson, 2006). The consolidation of a belief base B is denoted $B!$. It can be modeled as a contraction by the contradictory belief $\alpha_{\perp}$(Hansson & Wassermann, 2002), i.e. $B! = B - \alpha_{\perp}$. In what follows we introduce two operators for the base consolidation: the partial meet consolidation and the kernel consolidation as well as the set of postulates that characterize each operator.

Partial meet consolidation

The partial meet consolidation is a partial meet contraction by the contradictory belief.

Definition 2.9 (Partial Meet Consolidation): let B a belief base and a selection function γ, the partial meet consolidation of B, is the operator defined by:

$$B!_{\gamma} = \bigcap \gamma(B \perp \alpha_{\perp})$$

The following theorem characterizes the partial meet consolidation operator (Hansson & Wassermann, 2002).

Theorem 2.4 (Partial Meet Consolidation Representation): the operator ! is a partial meet consolidation for a belief base B if and only if it satisfies the following postulates:

[Consistency] $B! \nvDash \alpha_{\perp}$

[Inclusion] $B! \subseteq B$

[Relevance] if $\beta \in B$ and $\beta \notin B!$, then there is a subset B' of B such that, $B! \subseteq B' \subseteq B$ $B' \nvDash \alpha_{\perp}$ but $B' \sqcup \{\beta\} \vDash \alpha_{\perp}$

Kernel consolidation

The kernel consolidation is a kernel contraction by the contradictory belief. For each inconsistency element of the kernel, the consolidation removes from the

belief base at least one element that is responsible for this inconsistency. Formally,

Definition 2.10 (Kernel Consolidation): let B a belief base and σ an incision function, the kernel consolidation of B is the operator defined as:

$B!_{\sigma} = B \setminus \sigma(B \parallel \alpha_{\perp})$

The following theorem characterizes the kernel consolidation operator (Hansson & Wassermann, 2002).

Theorem 2.5 (Kernel Consolidation Representation theorem): the operator ! is a kernel consolidation for a belief base B if and only if it satisfies the following postulates:

[Consistency] $B! \nvDash \alpha_{\perp}$

[Inclusion] $B! \subseteq B$

[Core − retainment] if $\beta \in B$ and $\beta \notin B!$, then there is a subset B' of B such that, $B' \nvDash \perp$ but $B' \sqcup \{\beta\} \vDash \alpha_{\perp}$

2.3 Diagnosis Theory

Diagnostic reasoning systems have known two different approaches (Reiter, 1987). In the first approach, often referred as a diagnosis from the first principle, a diagnosis task is normally defined in terms of a set of components in which a fault might have occurred, a system description defining the behavior of the system, and a set of observations (or symptoms). If there is a discrepancy between the behavior and observations, a diagnose agent should determine the subset of components which may be the responsible. The only information available for him to solve the problem is system description. Under the second approach, the structure of the system is weakly represented and heuristic information plays an important role in the diagnosis task. The diagnosis task relies on the codified experience of the human expert being modelled, rather than on the deep knowledge of the system being diagnosed. We focus in this section on the first approach.

In order to seek a very general diagnosis theory, Reiter (1987) used the first-order language to represent task specific information.

Definition 2.11 (System description): a system is a pair $(SD, COMP)$, where

1) SD, the system description, is a set of first-order sentences.

2) $COMP$, the system components, is a finite set of constants.

The system description shows the normal behavior of system components. In all applications, the system description should mention a distinguishable unary predicate AB (Abnormal) to mean an abnormal behavior. Real world applications integrate observations in order to control components behavior. An observation is a finite set of first-order sentences. A diagnosis is called only if a discrepancy between the observation and the system description appears.

Definition 2.12 (Diagnosis): A diagnosis for a system with an observation OBS, $(SD, COMP, OBS)$ is a minimal subset $\Delta \subseteq COMP$, such that

$SD \cup OBS \cup \{AB(c) | c \in \Delta\} \cup \{\neg AB(c) | c \in COMP - \Delta\}$ is consistent.

Hence, 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.

In order to compute diagnosis, Reiter (1987) proposes a method based upon a suitable formalization of the concept of a conflict set. A conflict set is a subset of the system components that together produce an abnormal behavior.

Definition 2.13 (Conflict set): a conflict set for $(SD, COMP, OBS)$ is a subset of components $\{c_1, c_2, \dots, c_k\} \subseteq COMP$ such that

$SD \cup OBS \cup \{\neg AB(c_1), \neg AB(c_2), \dots, \neg AB(c_k)\}$ is inconsistent

A conflict set is minimal if no proper subset of it is a conflict set. The minimal conflict set presents the advantage to repair the problem of diagnosis by fixing one element in the set. For that purpose, Reiter (1987) introduced the notion of Hitting set:

Definition 2.14 (Hitting set): Given a collection $C$of sets, a Hitting set of C is a set H such that $H \subseteq \bigcup_{S \in C} S$ for $S \in C$and $H \cap S = \phi$. A Hitting set for C is minimal if there is no proper subset of it that is a Hitting set for C.

Since, the same symptom can caused by different conflict sets, the set of these conflict set constitute the collection C and the diagnosis should be the minimal Hitting set of it. This is the main result of diagnosis theory from the first principle as it was argued by Reiter (1987).

Theorem 2.6 (Diagnosis): $\Delta \subseteq COMP$ is a diagnosis for $(SD, COMP, OBS)$ if and only if Δ is a minimal Hitting set for the collection of conflict sets for $(SD, COMP, OBS)$.

Computing all diagnosis turns then to compute the set of minimal Hitting sets. For that purpose, Reiter (1987) proposed the following algorithm.

Definition 2.15 (Minimal Hitting sets Algorithm): Given a collection of sets F, an edge-labeled and node-labeled tree T is an HS-tree for the collection F if and only if it is the smallest tree such that,

(1) its root is labeled by $\surd$ if F is empty. Otherwise, it is labeled by an arbitrary set of F.

(2) If n is a node of the tree T, define H(n) to be the set of edge labels on the path from the root to the node n. if n is labeled by $\surd$, it has no successor nodes in the tree. If n is labeled by a set Σ of F, then for each $\sigma \in \Sigma$, n has a successor node n_σ joined to n by an edge labeled by σ. The label for n_σ is a set $S \in F$ such that $S \cap H(n) = \phi$. If such a set S exists. Otherwise, n_σ is labeled by $\surd$.

2.4 Conclusion

In this chapter, we presented in a nutshell separately both theories of belief revision and diagnosis theory. The former is a philosophical discipline, whereas the latter is originated from the artificial intelligence discipline. The AGM model is the most influential work in belief revision theory. It was originally derived from legal theory where beliefs are considered as closed sets under the logical

consequence relation. The model has been extended beyond this theory to meet new fields such as database updates and knowledge engineering. This practical mutation was the direct impact that gave birth to the new model of base revision theory. The AGM and base revision models rely on a nice and a precise logical formalism to define a variety of change operators each of which is characterized by a set of postulates to constraint the performed change. However, both models lack the implementation of realistic revision methods. Diagnosis theory from first principle is another field of research with the objective to restore systems consistency after deficiencies. The research field of diagnosis theory has powerful tools to prune computational complexity to the problem of diagnosis computing, allowing them to deal with real-world situations. The Hitting set algorithm is the main result of this theory for computing diagnosis. In the next chapters of this dissertation, we show how to applicate techniques from both theories of base revision and diagnosis from first principle to resolve the alignment evolution problem.

Chapter 3. Alignment change: The state of the art

3.1 Introduction

Usually, alignment is subject to change during its life cycle. Many reasons can trigger this change. Alignments cannot keep their consistency in time because of the dynamicity of ontologies. For instance, adding new knowledge in ontologies can make alignment inconsistent (Euzenat, 2015). Retracting knowledge from ontologies in response to some needs forces also alignment to follow this change. Often, created alignments are incomplete which may not satisfy all applications needs. Alignment can be improved manually or automatically by adding some correspondences which may cause alignment inconsistency. Hence, alignment needs to be evolved and maintained in order to follow the change and restore consistency. Another reason that can trigger alignment change is alignment debugging and repair. Ontology matching tools may produce erroneous correspondences that can lead to an inconsistent alignment as well (Meilicke & Stuckenschmisdt, 2009; Meilicke et al., 2007; Qi et al., 2009). The matching process should be followed by another process of debugging and repair to determine and correct faults.

Since the ontology evolution frameworks (Stojanovic, 2004; Plessers, 2006; Klein, 2004) integrate the evolution of depending artifacts as a particular task in the ontology evolution process we review in section 3.3 the main works of this field of research. Then we discuss their applicability for the case of the alignment evolution problem in section 3.4. Before that we should first clarify the notion of ontologies as it is used in computer science in section 3.2. Similarly, we present the notion of ontology alignment in section 3.5 followed by its life cycle in section 3.6. We review the state of art of the alignment change approaches in section 3.7. We classify these approaches and we discuss their outcomes according to the already fixed requirements in the introduction of this thesis. We conclude the chapter in section 3.8.

3.2 Ontologies

Ontology (with a big O) is a branch of philosophy for studying the nature and identities of things. In this discipline, philosophers try to answer questions concerning what things exist, which attributes characterize them and how such things can be grouped. Transferred to Artificial Intelligence, ontologies (with a little o) are computational artifacts that symbolize a special kind of knowledge. According to Gruber (1993), “for AI system, what exist is that which can be represented”. So, an ontology specifies explicitly the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. This was behind the Gruber definition of an ontology “explicit specification of a conceptualization” (Gruber, 1993). An explicit specification of a conceptualization can be done extensionally or intentionally (Guarino et al., 2009). The extensional specification is listing all possible interpretations of the vocabulary elements used by the conceptualization to name its elements. This is not always possible if the universe of discourse or the set of possible interpretations is infinite. However, the intentional specification constrains the intended meaning of the vocabulary elements by using a set of suitable axioms. The set of such axioms capture the intended interpretations corresponding to the specified conceptualization and exclude the unintended ones.

In summary, an ontology is an axiomatization of the intended meaning of a vocabulary used by a conceptualization of some area of interest. The manner of the axiomatization had led Uschold and Gruninger (2004) to give a continuum of kinds of ontologies (Figure 2). The spectrum expresses the meaning expressiveness as well as the formality of ontologies that increase from left to right. We qualify the two poles of the spectrum by "weak meaning" and "strong meaning" respectively. On the weak side, we can express a very simple meaning; on the strong side, we can express an arbitrary and complex meaning. 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).

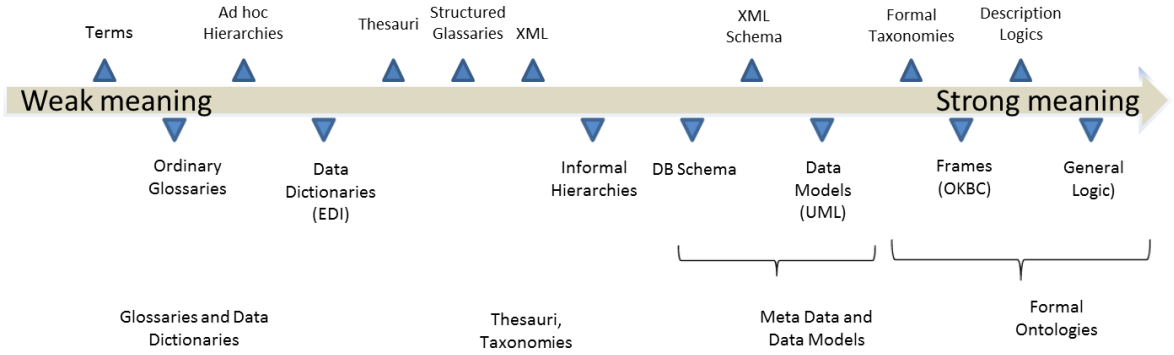

Figure 2: Ontologies spectrum

In order to meet semantic interoperability goals, ontologies should express a shared view of the domain of knowledge rather an individual view. The specification as a set of axioms can be given in informal, semi-formal or formal languages. If we want to extend semantic interoperability to encompass machines, ontologies should be formal. Studer et al. (1998) redefine the ontology as follows: "An ontology is a formal, explicit specification of a shared conceptualization". Basically, formal ontologies use languages that underlay different knowledge representation paradigms. Traditional languages such as Cycl (Lenat & Guha, 1989), KIF (Genesereth &Fikes, 1992), Ontololingua (Farquhar et al., 1997), Flogic (Kifer et al., 1995) are based on frames combined with first order logic. Classic (Patel-Schneider et al., 1991) and LOOM (MacGregor, 1991) are based on description logics. With the advent of the semantic web, new ontology languages have emerged. RDF (Lassila and Swick, 1999) is based on semantic networks to describe web resources. RDFS (Brickley and Guha, 2004) add frame primitives to RDF to organize web metadata. OWL (Dean et al., 2004) was built on top of RDF(S). OWL include some features from frames and others from description logics to specify more explicitly vocabulary meaning.

Semantic networks use directed graphs to represent individuals, objects, and abstract classes as vertices and semantic relations as edges. Within frames paradigm (Brachman & Levesque, 1992), we can view the world as frames (classes). A frame is just a list of slots (attributes). Values that can be assigned to slots are called fillers. Two particular slots are recognized in frames paradigm:

instance-of and is-a. The slot instance-of relates an individual frame to a generic frame. The slot is-a organizes generic frames in a hierarchy structure that take inheritance as the main reasoning principle. A generic frame inherits slots and fillers from more generic frames. Exceptions hold when the same slot is attached to both frames but with different fillers. This is why frame reasoning is qualified as non-monotone. Description logics (Baader & Nutt, 2003) view the world as concepts which are sets of individuals that play roles. Roles are modeling relationships between individuals. Subsumption is a particular relation between concepts modeling generality between concepts. The more general concept subsumes the specialized one. The main reasoning services for description logics are consistency checking, computation of the taxonomy, testing for unsatisfiable concepts, and instance retrieval. Unlike frames, reasoning in description logics is monotone.

3.3 Ontology Change

3.3.1 Origins

As was mentioned earlier, an ontology is an explicit specification of a conceptualization of a domain. An ontology change may occur following the change of any element of this definition (Klein & Fensel 2001; Noy & Klein, 2004). Namely, the described domain, the conceptualization of this domain and its specification are subjects of change. Changes in domains are not rare. For instance, domains merging change the initial domains to their fusion. A change in viewpoints and the usage of ontologies are some of many reasons that lead to changes in the conceptualization. In turn, the specification change can occur when translating an ontology from a language to another. A more expressive language allows representing more explicitly implicit knowledge.

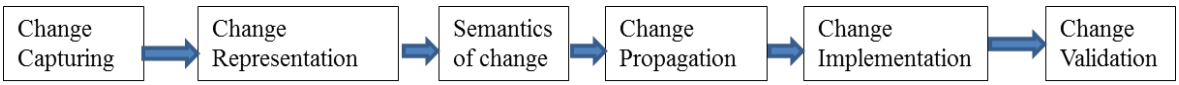

Figure 3: Six-phase ontology evolution process

3.3.2 Change process

Ontology change is the result of an evolution and or a repair process. According to Stojanovic (2004), “Ontology Evolution is the timely adaptation of an ontology to the arisen changes and the consistent propagation of these changes to dependent artifacts”. Since the change can affect other parts of the ontology as well as dependent artifacts, she proposed a process to execute the change. This process encompasses six phases: a phase for change capturing, a phase for change representation, a phase for the semantics of change, a phase for change propagation, a phase for change implementation, and a phase for change validation. Figure 3 schematizes the different phases of the ontology evolution process. During the first phase, changes are captured either from explicit requirements or from the result of change discovery methods. In the change representation phase, the captured change should be explicitly represented. Applying change may introduce inconsistencies in the ontology. Thus, the main concern of the semantics of change phase is to resolve these inconsistencies in order to bring the ontology in a consistent state. The semantics of change should be propagated to dependent artifacts as well. This is the task of the change propagation phase. During the implementation phase, ontology engineer should be informed about all consequences of any change request before changes are applied. Usually, ontology change constitutes the trace of evolution which may be used in the rollback to the old ontology. In this case, the ontology change is explicit and often stored in which is called a journal of change. Finally, the change validation phase enables justification of performed changes and undoing them at user’s request. It has as purpose to increase the usability of the evolution process. Change propagation of depending ontologies should follow the same change process as the single ontology evolution change process model.

In general, the change propagation can’t hold for many reasons (Plessers, 2006). The web is distributed and decentralized environment. Hence, we are usually unaware of all depending artifacts of an ontology. Propagating changes to them turns out to be even more problematic. Maintenance of ontologies and their depending artifacts is not necessary delegated to the same persons. Consequently, ontology engineer may have no permission to modify a given dependent artifact. Furthermore, maintainers may not want to update artifacts at that moment or at all.

For these reasons, Plessers (2006) distinguish between the evolution of an ontology and their depending artifacts. The evolution of an ontology holds following a request of change while the evolution of an artifact is a response to an ontology change. The former executes the evolution process of an ontology in five phases: change request, change maintenance, change detection, change recovery, and change implementation while the latter performs the evolution process of a depending artifact in three phases: a phase for change detection, a phase for the cost of evolution analysis, and a phase for analysis of versions consistency. Figure 4 shows an overview of the different phases for both tasks of the ontology evolution process. Ontology engineers express their request for change during the change request phase. Check and resolving inconsistencies by introducing deduced changes are the main concern of the consistency maintenance phase. In order to provide a better understanding of the evolution of an ontology, changes that were not explicitly listed in the change request are detected during this phase. Unnecessary deduced changes can be recovered during the change recovery phase. The goal of change implementation phase is to implement the requested and deduced changes into an actual ontology. The change detection phase for the evolution in response task allows maintainers of dependent artifacts to create their own set of change regardless of the detected change during the evolution in request task. The goal of the cost of evolution phase is to reveal the inconsistencies that an update would introduce and to indicate the latest backward compatible version. The purpose of the version consistency phase is to keep a depending artifact consistent either by updating it with a backward compatible version or not by the maintainer.

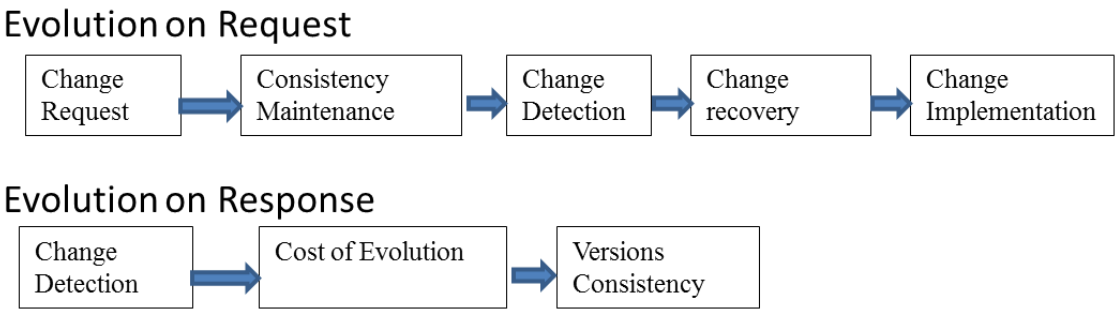

Figure 4: On request and on response ontology evolution processes

Schema evolution provides both access to the old and the new data via the new schema and schema versioning permit both access to old and new data from

different schema interfaces. In open environment such as the semantic web, this traditional distinction is no longer applicable to ontologies. This means multiple versions of the same ontology are bound to exist and must be supported. Dependent artifacts of an ontology are free either to move to the new version or to keep the connection with the old one. These leaded Klein (2004) to consider the change management as the key issue in the support of ontology evolution. Hence, he combined the ontology evolution and versioning in the same concept defined by "the ability to manage ontology changes and their effects by creating and maintaining different variants of the ontology". Thus, the focus of attention of a versioning management system is to represent the change as well as to specify mappings, perform transformation, comparison, and identify compatibility between versions of the same ontology. Ontology change in this setting is always implicit and embedded in versions rather than in journals as in evolution environment. It is the role of management systems to compute the change and make it available for applications so that they can update their functionalities.

In the next section, we focus on some approaches that instantiate some or all phases of these process of change.

(a) Change Representation

Ontology change is a set of operations applied to ontology elements. Two types of operations are distinguished in the literature: elementary operations and composite operations. Elementary operations can only apply to one ontology element while composite operations are the composition of such elementary operations and other composite operations. The expressiveness of change representation is directly influenced by the expressiveness of the ontology language. For instance, Stojanovic (2004) consider KAON ontologies. KAON language is based on an extension of RDF(S). She distinguished three types of change operations: elementary change, composite change, and complex change. Elementary change is a basic operation that modifies an ontology element. Composite change is any change that modifies elements neighbors and a complex change is any change that can be decomposed into any combination of at least two elementary and composite ontology changes. This classification of the ontology change forms an ontology of change. This ontology models what changes, why, when, by whom, and how are performed in an ontology. The main concepts of this

ontology are additive change and contractive change. Additive change is to add a new element to an ontology. Contractive change is to remove an element from the ontology. A change log uses the vocabulary of this ontology to store change as an instance of this ontology. Klein (2004) considers OWL ontologies which are more expressive than KAON ontologies. He followed the same change classification as the previous approach with the exception of basics operations where he includes the modify operation as a subclass of this category of change although it can be considered as the composition of a contractive followed by an additive change. Elementary changes are called atomic operations and a composite change consists of a composition of any set of atomic operations. Unlike the previous approach, this ontology of change contains more types of change relatively to OWL, the underlying ontology language.

Unlike to previous approaches, Palma et al. (2009) propose a generic ontology for the representation of ontology changes. This ontology models generic operations as taxonomy of changes that are expected to be supported by any ontology language. The proposed taxonomy extends previous approaches with a more granular classification which considers the actual atomic changes that can be performed upon an ontology. Moreover, the taxonomy can be specialized for different ontology languages.

(b) Change detection

The change detection allows ontology engineers to create their own set of change by comparing versions. The comparison between versions has been the subject of several approaches (Noy & Musen, 2002; Klein et al., 2002; Papavassiliou et al., 2009; Redmond & Noy, 2011; Kremen et al., 2011; Hartung et al., 2013). PromptDiff (Noy & Musen (2002) and COnto-Diff (Hartung et al., 2013) consider directed acyclic graphs like ontologies. They support the detection of several basic as well as complex changes including concept additions, deletions, splits and merges. Both approaches achieve comparison in two steps: version matching and a structural difference computing. The PormptDiff algorithm (Noy & Musen, 2002) uses an extensible set of heuristic matchers (e.g., single unmatched sibling, unmatched inverse slots, or same type/name) to detect changes between two ontology versions and a fixed-point algorithm to combine the results of matchers to produce a structural diff between them. COnto-Diff (Hartung et al.,

2013) uses the system GOMMA (Kirsten et al., 2011) to determine corresponding concepts in the input ontology versions and then it applies a set of rules to detect ontology changes.

OntoView (Klein et al., 2002) and Papavassiliou et al (2009) consider RDF triple-based ontologies. The OntoView algorithm (Klein et al., 2002) uses a graph representation as well as a set of rules to detect basic changes between versions. Complex changes such as merges or splits are not supported. Papavassiliou et al (2009) distinguishes between low level changes and high level changes between versions of RDF(S) ontologies. Low level changes are the set of added and deleted RDF triples. High level changes correspond to basic and composite changes. Basic changes describe a change in one node or edge of the graph corresponding to the RDF(S) ontology while composite changes describe changes affecting several nodes and/or edges of the RDF(S) ontology. In order to insure determinism in change detection, composite changes takes precedence over the detection of basic ones.

A recent approach (Redmond & Noy, 2011) inspired by PromptDiff (Noy & Musen, 2002) was born out of the need to support OWL 2. It determines the difference between the signatures and axioms of the two versions and reorganizes the axiom changes into a format that is more readable to a human. Unlike this approach, another tool (Kremen et al., 2011) computes the difference between OWL ontologies only as a set of axioms and it does not consider separately the difference of signatures. While the former consider a purely structural difference, the latter compute the logical difference by taking a more deep analysis to check among the changed axioms that still entailed.

(c) Consistency checking and resolution

An ontology is consistent if it satisfies some consistent conditions. Haas and Stojanovic (2005) distinguish three types of consistency: structural, logical and user defined consistency. The structural consistency depends on the underlying models of ontologies. Conditions of consistency are constraints that are defined for the ontology model with respect to constructs that are allowed to form the elements of an ontology. While the structural consistency is determined by a set of conditions the logical consistency ensures no contradiction can be entailed from

ontologies. The user-defined consistency refers to user requirements that need to be expressed outside of the ontology language itself. In order to resolve inconsistencies, additional change should be generated to bring the ontology in a consistent state. They introduced different ways baptized strategies to resolve every particular inconsistency. Different strategies may generate different additional change and hence different ontologies.

While works on schema evolution have largely influenced the resolution of the structural consistency, the theory of belief revision was the main inspiration of the works of solving the logical inconsistency problem. In belief revision theory, inconsistency is only one postulate from others that can constrain an evolutional operator (Alchourrón et al. 1985). Some results related to the feasibility of applying AGM model for ontology evolution have appeared in (Flouris 2006). The author showed that some description logics cannot satisfy all the AGM postulates. These logics are not AGM-compliant. The absence of negation is one of many problems that hamper the application of AGM model in description logics. Ribeiro and Wassermann (2007) have undertaken the problem of belief revision for logics without negation by adapting kernel operators (Hansson, 1994). A base kernel is a set of all minimal subset of a belief base that causes the entailment of some sentences. One advantage of the base kernel is the simply removing of one element from each set can stop the entailment. See chapter 2, for more details on this point.

Justification is the same idea as the base kernel. It is an important notion for explanations in ontologies (Horridge, 2011). Some errors such as contradictions may occur following the design of ontologies which may hamper their usage. Ontology debugging and repair is the process of diagnosing and repairing such errors (Kalyanpur, 2006). The justification, a minimal subset of an ontology that is sufficient for an entailment to hold, constitutes a good mean for explanations in ontology debugging and repair research field.

(d) Change propagation

Klein (2004) shows how to deal with the tasks of ontology synchronization as well as the integrity of modular ontologies problem in his framework. The ontology synchronization introduced by Oliver (2000) consists of change

propagation from a shared vocabulary to a local vocabulary in his framework. A modular ontology refers to concepts of others ontologies. The integrity of a modular ontology aims at keeping the local reasoning valid even the external referred concepts are changed. Stojanovic (2004) studied change propagation to dependent ontologies. In this approach, ontologies depend on replications instead of their original ontologies. She investigated two ways of propagation. The former is called Push-based approach. Changes are propagated as they happen to dependent ontologies that reside on the same node as the original ontology. The push-based approach which is suitable when strict dependent ontology consistency is required since the information about the original state of the changed ontology is available for the evolution of the dependent ontology. The latter is the pull-based approach. Changes are propagated to distributed dependent ontologies residing at a different node of the network only at their explicit request. The pull-based approach is better suited for less stringent consistency requirements. This technique of propagation might work for a controlled environment where ontologies fall under the same authority. However, in uncontrolled setting ontology engineers cannot be forced to update to the latest version of an ontology they depend on but they can choose between deferring changes, updating to a backward compatible, or to a non-backward compatible version of an ontology (Plessers, 2006). To resolve the inconsistency, Plessers (2006) proposed the rupture of dependencies between ontologies and copying the essentials entities from changed ontologies to dependent one.

3.3.3 Change process support

Nowadays, many ontology development tools exist. It is rare to found a tool that supports completely the ontology evolution process. KAON implements the entire six phases of change process (Stojanovic , 2004). Protégé (Noy et al. 2000) is one of the most popular tools for ontology design and creation. The functionality of Protégé has been enhanced so as to provide several interesting features useful for both ontology design and evolution (Noy et al.2006). Two new plugins for change handling come to join the set of protégé plugins. The Change-management plugin provides access to a list of changes and enables users to add annotations to individual changes or groups of changes; when this plugin is activated, the

changes are stored as instances in an ontology of change. This ontology extends a previously developed ontology jointly with Klein (2004). The Prmpt (Noy & Musen., 2003) plugin is a suite of tools for ontology management. Besides, versions comparison, it provides facilities to examine a list of users who performed changes and to accept and reject changes.

The NeOn is a kit of ontology development tools. The development of ontologies is considered within a network of ontologies (Haase et al. 2006), defined as a collection of ontologies related together via a variety of different relationships such as mapping, modularization, version, and dependency relationships. NeOn permits the manual application of change to ontologies. It contains a list of plugins such that each of them supports one aspect or more of the ontology evolution process. The change capturing plugin supports the logging of changes automatically from the NeOn ontology editor. It also supports the application of logs generated by other systems. Additionally, it is also in charge of propagating changes to the distributed copies of the same ontology. Evolva is a plugin that supports change discovery from external data sources such as text, folksonomies, and RSS feeds. RaDon supports change inconsistencies diagnosis and repair. Finally, NeOn permits change verification and validation.

3.4 Discussion

Ontology evolution approaches don’t study the alignment evolution as a particular depending artifact. Instead, they describe general frameworks for the evolution of depending artifacts regardless of their natures. In what follows we discuss the applicability of these frameworks to the problem of alignment evolution under ontology change. The discussion will be guided by the already introduced requirements of alignment evolution problem, namely, the ontology change identification, the consistency, the minimality of change, and users’ involvement. First, we discuss globally how ontology evolution frameworks can embed these requirements in their change process. Then we discuss their outcomes for every requirement.

a) The change process: Stojanovic (2004) studied the propagation of an ontology change to dependent ontologies. Change propagation of dependent ontologies

should follow the same change process as the single ontology evolution change process model (See figure 3). This process model can also be adapted for alignment evolution problem. The change capturing phase can be refined to a change identification phase. The change representation phase should be a part of the change identification phase since the identification can't hold without the consideration of some patterns of change. The semantics of change phase ensures the consistency of alignments following ontology change. In huge, distributed, and decentralized environments such as the semantic web we can't aware of all applications depending to this alignment. Hence, we envisage two types of propagation. A push-based approach that propagates alignment changes to applications which are managed by the same maintainer as the alignment. A pull-based approach that propagates changes to applications which are not under the authority of the alignment maintainer. But we can satisfied by delivering an evolution log as a journal of change to these applications. Alignment evolution log shows the difference between the old and the new alignment. This is the role of the change implementation phase. Before that, the alignment maintainer should validate the change. The system shows the inconsistencies, gives explanations, and proposes changes. In turn, alignments maintainers validate the change, recover the unnecessary change, adapt it, or cancel the change by keeping connection with the old version of the evolved ontology if it is available.

Plessers (2006) differentiates the change of ontologies from the change of their dependent artifacts. He proposed a process of three phases for artifacts evolution (See figure 4). The requirement of change identification can be fulfilled during the phase of change detection. During this phase, the alignment evolution system should offer an interface at the side of the alignment maintainer which allows him to detect and represent changes according to its own vision. The requirements of the consistency change and minimal change can be the subject of the phase of the evolution cost. During this phase, the system shows inconsistencies, gives explanations, and proposes changes. This assistance serves as a guide for maintainers to decide or not the change. The main purpose of versions consistency phase is helping maintainers to find previous backward compatible versions of the changed ontology since they are free to choose updating or not their artifacts. This helps change tracking by alignments maintainers. The maintainer can accept the

change, recover the unnecessary changes, adapt the change before updating the alignment or cancel it if an old backward compatible version is available. Like ontologies, alignments have dependent applications that should be updated following their changes. The alignment evolution system should implement and deliver the alignment change to these applications. Fortunately, the alignment change is not as complex as the ontology change and it can be easily understand by applications maintainers. Hence, applications maintainers don't need to create their own sets of change. However, the delivered set helps them to evaluate the cost of evolution, to check backward compatibility, and to decide updating or not their applications. Also, versions consistency can facilitate to retrieve backward compatible versions of alignments.

In summary, both discussed ontology evolution frameworks should be adapted to fulfil the cited requirements. As ontologies, alignment evolution has its own depending applications which may create a confuse evolution with artifacts of evolved ontologies. Consequently, we think separating the task of ontology evolution from that of alignment evolution helps in removing such confusion.

b) Change representation and detection: the change representation task aims at drawing the set of patterns of change relating to a model of an evolved ontology. While the change detection task match between an ontology change and these patterns of change. These patterns of change constitute ontologies of change. However, these ontologies of change are ontology languages dependent which hamper them to reach the wished consensus. Nevertheless, these ontologies and detection tools of change may constitute a library for alignments maintainers to choose between them according to their needs.

c) Checking and resolving inconsistencies: the ontology inconsistency expresses a set of hard constraints which condition the usefulness of ontologies. Checking inconsistencies turns to checking the violation of at least one of these constraints. Different ways may exist for resolving the same inconsistency. One criterion that can guide the resolution is the minimal change. Unlike consistency, the minimal change expresses a soft constraint since it doesn't affect the usefulness of ontologies. Which means; consistency constraints take precedence over minimal change during the resolution of inconsistency. Sometimes, it is inevitable to sacrifice the minimal change against the consistency satisfaction. The challenge

question is how to ensure the compromise between the consistency and the minimal of change constraints. Among all the above presented approaches, only belief revision based approaches can reach this objective. In Belief revision theory, a change is a set of rational operators constrained by a set of postulates (See chapter 2 for more details). A promise investigation is how to apply this theory for the alignment evolution problem as well. The alignment evolution is a knowledge intensive task which needs users' involvements. Justifications as explanations approaches can also play big roles for the alignment evolution problem.

3.5 Ontology alignment

According to Guarino et al (2009), an ontology can only approximate the specification of a conceptualization and the degree of such specification depends (1) on the richness of the universe of discourse (2) on the richness of the vocabulary chosen (3) on the axiomatization. This divergence in the vocabulary chosen as well as in its axiomatization also called terminology heterogeneity and conceptual heterogeneity respectively (Euzenat & shvaiko, 2013) may lead to the development of heterogenic ontologies of the same universe of discourse. Overlapping universes of discourses may lead to overlapping ontologies as well. Relating ontologies by stating semantic relations between their vocabularies constitutes which is called an ontology alignment. A semantic relation expresses how meanings of both vocabularies are related. Usually, the set-theoretical relations are used to specify such relations. The equivalence relation expresses related meanings are the same, the inclusion relation expresses meanings inclusion, the overlapping relation expresses meanings overlapping, and the exclusion relation expresses meanings disjointness. Even, a mapping should be a mathematical function whereas an alignment is a general semantic relation between ontologies; some authors (Kalfoglou & Schorlemmer, 2003; Noy, 2009) use the term mapping instead of alignment.

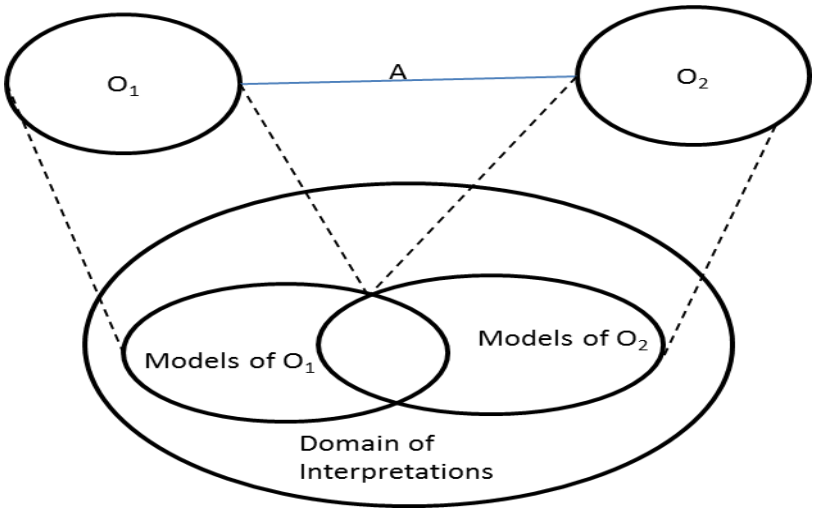

Figure 5: Model theoretic based alignment global semantics

Since alignments relate ontologies, interpretations of semantic relations should remain compatible with interpretations of related vocabularies. In other words, alignment should not impose new interpretations for ontologies or change the previous one but only show how interpretations are related. We distinguish two approaches to relate interpretations of ontologies relatively to the domains of interpretations. When ontologies describe the same domain of interpretation, an alignment interpretation becomes a part of the global interpretation formed by the union of the different interpretations of the aligned ontologies. This is informally presented in Figure 5. In this approach, the aligned ontologies together with the alignment form a global ontology. We can use OWL constructs to express alignments between entities of the different ontologies. The construct owl:import allows to import all entities of the aligned ontologies in the space of the global ontology and constructs such as owl:equivalentClass, rdfs:subclassOf, and owl:disjointClasses permit to represent set-theoretical relations between them. For contextual interpretations which reflect different points of view on the same real world entities, semantic relations of an ontology alignment are interpreted as bridge rules relating these interpretations (Bouquet et al, 2003). These rules express how to translate instances from the source ontology to the target ontology. The C-OWL language extends OWL by embedding bridge rules to represent contextual alignments. Unlike OWL, C-OWL makes a clear separation between alignments and ontologies. Figure 6 shows intuitive interpretations of some bridge rules.

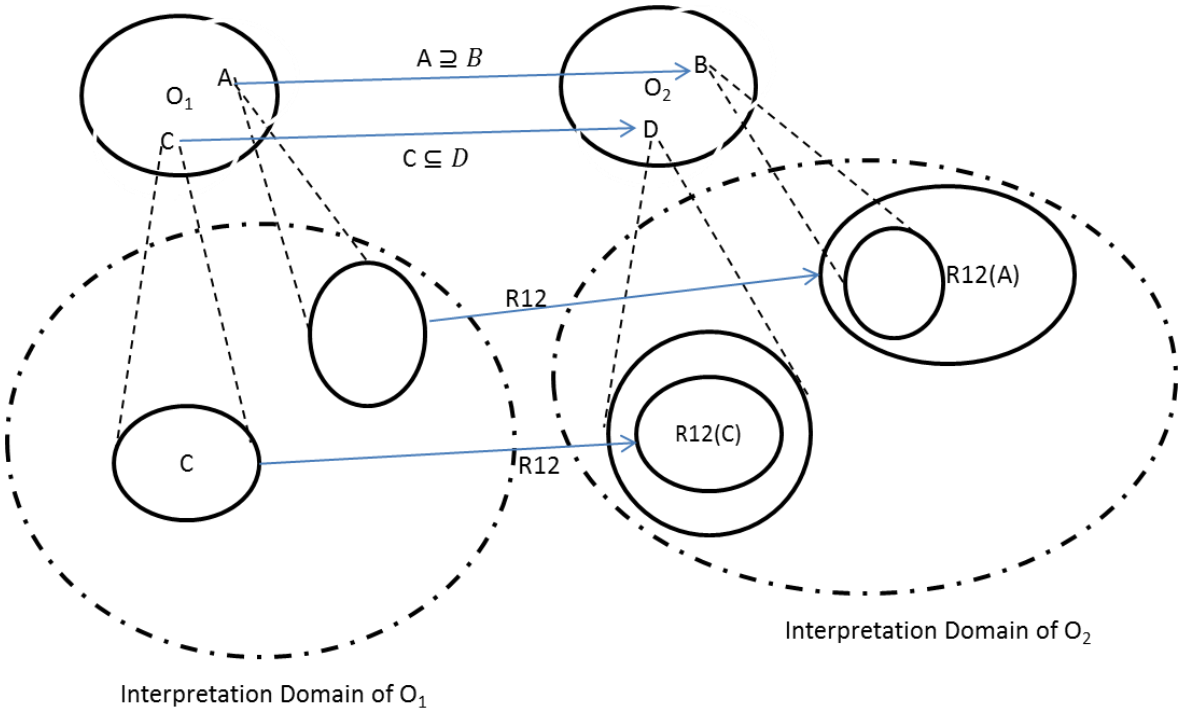

Figure 6: Model theoretic based alignment contextual semantics

The above-cited approaches rely on model theoretic semantics to give an extensional interpretation for alignments. The model theoretic semantics expresses extensionally how meanings of two different vocabularies are related. An alternative approach constraint intentionally the relations between meanings. Reductionist semantics (Meilicke & Stuckenschmidt, 2009) gives an axiomatization to constraint the alignment between entities in different ontologies. Within this semantics, an alignment interpreted as a set of axioms together with ontologies form a merged ontology. Reasoning on alignment turns to reasoning on this merged ontology. Nevertheless, the model theoretic semantics and the axiomatic semantics are not disjoint but we can move from the model theoretic semantics to the axiomatic one and vice versa. For instance, the alignment natural semantics which is a reductionist semantics where the semantic relations translated to axioms in some ontology language are joined to all axioms of both ontologies correspond to the model-theoretic semantics with one domain interpretation for all aligned ontologies.

MAFRA framework (Martins & Silva, 2009) gives an operational semantics to alignments by attaching services to their semantic relations. An alignment in this approach is an instance of an ontology named SBO (for Semantic Bridge

Ontology) to serve a representation and exchange mechanism of semantic relationships between ontologies. SBO specifies, interrelates, and classifies the types of alignment relations. Besides, the ontology provides other modelling constructs necessary to express alignments. The SBO ontology contains two main classes: SemanticBridge and Service. The SemanticBridge class in turn is specialized into two classes: ConceptBridge and PropertyBridge. The relation hasBridge associates a PropertyBridge to a ConceptBridge. The subBridgeOf relation gives a hierarchy structure to ConceptBridges. The ConceptBridge specifies a semantic relationship between source concepts of the first ontology with target concepts of the second ontology. The Service class implements the possibilities of transformation related to the class SemanticBridge. The service copy, always attached to the ConceptBridge, is responsible for translating instances of source concept to instances of the target concept during the execution phase.

3.6 Ontology alignment life cycle

Many tasks are related to the ontology alignment development and they are performed as long as its life cycle (Euzenat et al. 2008). We distinguish three main phases of this life cycle: The design phase, the sharing phase, and the using phase. Adapted from (Euzenat & shvaiko, 2013), figure 7 outlines these phases and their related tasks. The design phase is an iterative process formed by three tasks: the creation task, the evaluation task, and the enhancement task. The task of alignment creation known also by the ontology matching task is the first task in the life cycle which aims to create alignments. Nowadays, many performant ontology matching tools are available (Euzenat & shvaiko, 2013). Based on different aspects of the knowledge encoded in ontologies, they combine different techniques to match ontologies. For instance, terminological techniques compare the lexicon used to designate ontological entities. Some tools consider ontologies as directed acyclic graphs and hence they compare the structures that surround entities. In order to be useful, the obtained alignment should be evaluated. The evaluation task consists of assessing the correctness as well as the completeness of this alignment which might lead to an enhancement. The enhancement task 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. The tasks of creation, evaluation, and enhancement might then go through an iterative process until an alignment is deemed worth publishing. During the sharing phase, the alignment can be stored and communicated to other parties interested in such an alignment. Open servers are now available to store, index, organize and share alignments. For instance, Bioportal[8] is an open community-based repository of biomedical ontologies. Users can browse alignments, upload new alignments, and download alignments that the repository has (Noy et al. 2008). In the final phase, the alignment can be exploited. Servers can deliver the alignment in different formats in order to ensure its large usefulness. Then, applications interpret it according to their needs and using it to perform actions, like mediation and merging.

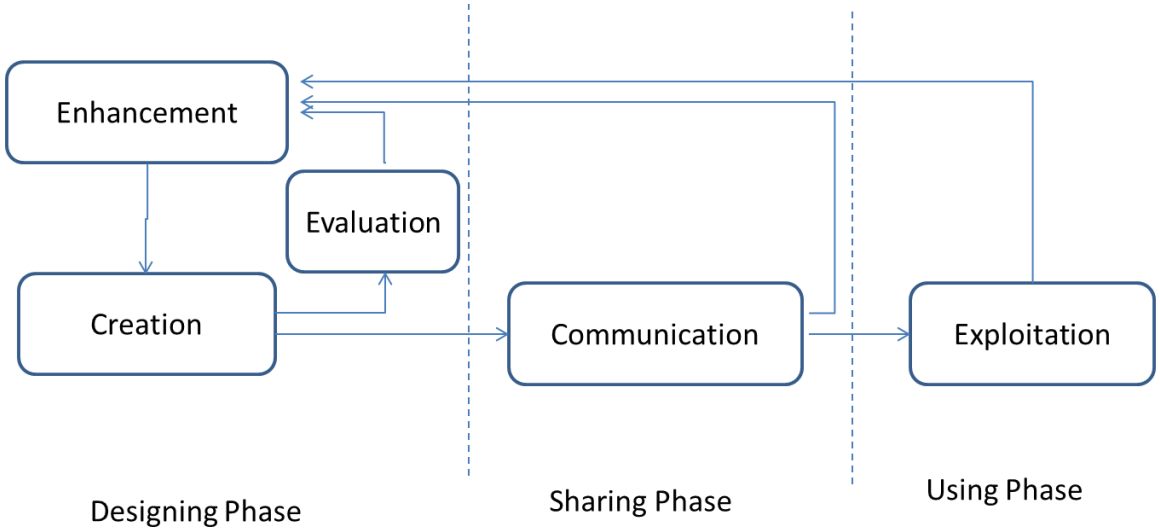

Figure 7: The ontology alignment life cycle

3.7 Alignment evolution

3.7.1 Naming disambiguation

In the literature, approaches studied the alignment evolution problem under various names: alignment adaptation, alignment maintenance, alignment evolution, and alignment revision (Dos Reis et al., 2015). Under the name alignment

[8] http://bioportal.bioontology.org

revision, Euzenat (2015) 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. This study considers ontologies as logical theories. Changing logical theories is the classical philosophy problem of belief revision where beliefs are logical theories (See chapter 2, for more details). While this approach borrowed the name of alignment revision from philosophy community other approaches (Groß et al., 2013; Dos Reis et al., 2013; Martins & Silva, 2009) follow the line of software engineering to adopt alignment evolution, alignment maintenance, and alignment adaptation names by considering ontologies and alignments as software products.

Alignment debugging is a task performed before alignment delivery to diagnose and repair alignment produced by ontology matching tools. Created alignments might contain errors such as redundancy, inconsistency, imprecision, or an abnormal behavior (Wang & Xu, 2008). Here again, alignment debugging knowns the same problem of naming ambiguity as alignment evolution problem. When a produced alignment between DL ontologies is inconsistent or incoherent, Meilicke et al (2009) and Qi et al (2009) study this problem under the name of alignment revision. By converting the alignment to a set of axioms and merging it with ontologies, they obtain a global knowledge base. Making consistent an inconsistent belief base is a particular operation in base revision theory (See chapter 2, for more details). Whereas, the name debugging is the most useful naming for programs debugging in software development domain, revision is the conventional name used for base change theory which leads these approaches adopting the name of revision instead of debugging name.

3.7.2 Classification

Software evolution and maintenance in software engineering is the set of activities which keep systems operational and meet user needs. Swanson (1976) identified three categories of maintenance: the corrective maintenance, the adaptive maintenance, and the perfective maintenance. The corrective maintenance is the reactive modification of a software product performed in response to the

assessment of failures. The adaptive maintenance is the modification of a software product performed in anticipation of change within the data or processing environments. The perfective maintenance is the modification of a software product performed to eliminate inefficiencies, enhance performance, or improve maintainability. The alignment maintenance task is of great importance as it aims to keep the alignment useful and ready in time during its life cycle. Alignment maintenance approaches follow different ways to maintain and evolve alignments. Some approaches view the problem of alignment evolution under ontology change as an adaptive process. The main challenge for them is how to modify alignments according to the detected changes in ontologies. Sometimes, ontology change such as adding concepts has no impact on alignment but designers prefer to extend it out of the need with new correspondences in order to enhance its usefulness. For some scenarios, this extension can be classified as a perfective maintenance. However, the same extension might become more than necessary for some applications who request a full interoperability and integration. In this scenario, the extension should be classified as an adaptive maintenance since it was born following a change in applications needs. Hence, we qualify this type of approaches as an adaptive and perfective maintenance. When an alignment extension or an ontology change hamper the usefulness of the alignment by introducing errors, other kind of approaches try to identify and correct these errors. We qualify this kind of approaches as a corrective maintenance. Similar techniques used by different approaches that aim to correct errors during alignment debugging can also be applicable for the alignment maintenance problem. Following this classification, we present and discuss the outcomes of these approaches. The requirements of the alignment evolution under ontology change problem fixed a priori in the introduction of this thesis are the main guide of this discussion. Table 1 summarizes this discussion.

3.7.2.1 Alignment adaptive and perfective maintenance

The main objective of adaptive maintenance approaches is adapting alignments according to changes in the implied ontologies. Approaches of this category (Groß et al, 2013; Dos Reis et al, 2013; Khattak et al, 2015) consider an ontology as a directed acyclic graph (DAG). They support the detection of several basic as well

as complex changes including concepts addition, deletion, split, merge, and move. Then, they use an ontology change handler to guide the alignment maintenance process that converts the change either to an alignment between the versions of the evolved ontology or to a set of actions that adapt the affected correspondences according to the type of change.

Groß et al. (2013) present two approaches for adapting the ontology alignment: the composition-based and diff-based adaptation approaches. Both approaches rely 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 using some functions such as the maximum or the aggregation to combine their associated semantics similarities. The composition based approach uses the ontology matching tool GOMMA (Kirsten et al., 2011) to convert the implicit ontology change represented by the presence of versions to an alignment while the diff-based approach converts every type of change to a semantic relation between entities concerned by this change. The diff-based approach uses COnto-Diff tool (Hartung et al., 2013) to identify basic changes like attribute value changes as well as complex change types such as concepts split or merge. Both approaches seek new match for added concepts with concepts of the target ontology to enhance the alignment with new correspondences. According to authors, the outcome of this adaptation process is a valid alignment. However, alignment validity is not explicitly defined but they let it to expert's appreciation. The correctness of the alignment composition depends on the correctness of the composed alignments. Both proposals rely on heuristic rules to generate an alignment between versions. Consequently, no guarantees are given to ensure the validity of the adapted alignment. Furthermore, the alignment composition is an incomplete method which might lead to unnecessary missing of some correspondences in the new alignment.

Dos Reis et al. (2013) present an automatic adaptation approach relying on a change handler which converts the change to mapping adaptation actions for adapting the affected correspondences according to the type of change. Based on the same tool COnto-Diff as the previous approaches, they compare versions and categorize changes according to a revision change, an addition change, or a

deletion change. They proposed five distinct mapping adaptation actions that represent different possibilities for adapting alignment: correspondences addition, correspondences remove, correspondences move, correspondences derivation, and modification of semantic relations. Remove and addition of correspondences are atomic actions while move, derive, and modification of semantic relations are composed actions. The move action re-allocates a correspondence in the alignment when it is judged invalid. The derivation action creates a modified copy of a correspondence which is still considered as valid. Usually, the modification action is used in conjunction with move and derivation action 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 an addition change. Alike 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.

Regardless of the change type, Khattak et al (2015) act by deleting all correspondences concerned by the change and then add new correspondences by partially re-computing the alignment. Exploring the change history log (Khattak et al, 2008) of the evolved ontologies, the approach reuses completely the unaffected part of the alignment and the changed elements in the source or the target ontology of the alignment are automatically matched with the complete current version of the other ontology. Without affecting precision, the proposed 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. The approach doesn’t much profit from the availability of the ontology change to adapt alignment. Instead, the approach uses changes only for filtering affected correspondences. Just seeking new match for changed entities can’t ensure alignment validity. This is why alignments produced by ontology

matching tools should be debugged to detect and correct erroneous correspondences (See section 3.7.2.3).

3.7.2.2 Alignment corrective maintenance

Unlike adaptive maintenance approaches, approaches of corrective maintenance detect and correct erroneous correspondences. Erroneous correspondences may be the consequences of the evolved ontologies or the evolution of the alignment by itself.

In (Martins & Silva, 2009), an alignment is an instance of Semantic Bridge Ontology (SBO). This ontology serves a representation and an exchange mechanism of semantic relationships between ontologies (See section 3.5). The evolution of alignment in this approach is a process that aims to preserve the semantics of this ontology when the deletion of concepts in the source or the target ontology is observed. Deletion of concepts leads to invalid entities of the ontology SBO such as invalid arguments for Concepts Bridge and Properties Bridge. Inspired by strategies applied in Stojanovic's ontology evolution framework (Stojanovic, 2004), they propose a list of strategies to correct invalid entities of SBO. In order to preserve as much as possible the old alignment, they sort the list of invalid entities. For instance, since Properties bridges are always defined in the context of Concepts Bridges, invalid Concepts bridges should be corrected first. Two methods are proposed to correct invalid entities. The first method is a 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 predicts the ontology evolution strategies from the journal of change. The changes are captured in a log that stores the exact sequence of changes made to update ontology. The authors establish a list of rules to identify the evolution of ontology scenario which determines the alignment evolution strategy. According to authors, ensuring SBO validity implies alignments validity. This is true at the structural level since valid entities in SBO always give valid correspondences in alignment. Moreover, this validity is given with a minimality of change by sorting invalid entities. These approaches study only the impact of deleted concepts on

alignments. It is not clear how they can ensure alignments validity for other types of change.

Euzenat (2015) 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. 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. Local inconsistencies may only be solved by local revision of the concerned ontology or the concerned alignment while these local operations can used independently to resolve the global inconsistency. The author considers an ontology as a logical theory (LT) formed by a set of axioms. According to him, the main problem in semantic web is to accumulate knowledge rather than to contract it. So, he focuses only on revision of the network with axioms. Mirroring the framework of AGM model of belief revision (Alchourrón et al., 1985), the approach introduces a set of postulates which constitute constraints to be fulfilled by any operator of local change on alignments as well as on ontologies. Then it provides postulates for revising the network of ontologies when an ontology of the network is revised by an axiom or an alignment is revised by a new correspondence. The approach showed that the global revision of the network is a generalization of local revisions. Besides, the approach defines a partial meet operator for the alignment revision that satisfies the fixed postulates. Unfortunately, this operator can't be in general a revision operator for the network. Instead a partial meet operator for a network can be designed as the intersection of selected maximal consistent sub-networks with respect to added axioms or correspondences. The approach provides alternatives 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 its worth do not modify them only if there is not another way. As it is mentioned by the author, this work can be considered as a first step to understand revision in networks of ontologies that may help to consider the problem of base revision. Belief sets in the AGM framework are closed sets under the logical consequence relation. While the

framework presents nice results it lacks practicability since closed sets are infinite or at least very large sets that cannot be incorporated easily into a computational framework (Peppas, 2008).

Table 1 : ***classification of alignment evolution approaches***

	Groß et al, 2013	Dos Reis et al, 2013	Khattak et al, 2015	Martins & Silva, 2009	Euzenat , 2015
Category	Adaptive and Perfective Maintenance			Corrective Maintenance	
Ontology Model	DAG	DAG	DAG	DAG	LT
Ontology Change	Basic and Complex	Basic and Complex	Basic and Complex	Basic	Basic
Alignment Model	Syntactic	Syntactic	Syntactic	Syntactic	Semantic
Alignment Consistency	Not defined	Not defined	Not Defined	Structural	Logical
Change Minimality	No	No	No	Yes	Yes
User Involvement	Adapting semantic relations	automatic	automatic	Choose strategies	automatic

3.7.2.3 Alignment debugging

Alignment produced by ontology matching tools may contain invalid correspondences. Qualified as invalid, every correspondence contributes in the violation of some defined constraints. Alignment debugging is the process of diagnosis and repair of such correspondences. Techniques used in alignment debugging can also be applicable in alignment maintenance, precisely, for the alignment corrective maintenance.

For some tools, the diagnosis of invalid correspondences is based on patterns of reasoning which are correct but incomplete reasoning methods. Lily (Wang & Xu,

2008) uses four types of patterns: redundant mapping, imprecise mapping, inconsistent mapping, and abnormal mapping. ASMOV (Jean-Mary et al., 2009) uses five types of patterns to check semantics: multiple-entity correspondences, crisscross correspondences, disjoint-subsumption contradiction, subsumption and equivalence incompleteness, domain and range incompleteness. The pattern disjoint-subsumption contradiction used by ASMOV corresponds to the inconsistent mapping pattern used by Lily. YAM++ (Ngo & Bellahsene, 2012) relies on ALCOMO[9] system to debug alignments. ALCOMO (Meilicke & Stuckenschmidt, 2007) uses disjoint-subsumption contradiction pattern to check the satisfiability preservation of entities by alignments.

Independent approaches (Meilicke & Stuckenschmidt , 2009) and (Qi et al, 2009) use techniques adapted from diagnosis and belief revision theories to establish the coherency of alignments between description logics based ontologies. The alignment coherency is a sort of logical consistency such that satisfiable ontological entities should preserve their satisfiability even when ontologies are connected by alignments. Both approaches use the notion of minimal conflict set to designate the minimal set of correspondences responsible for alignment incoherency. This set present the advantage to repair the problem of alignment incoherency by fixing one element in the set. Meilicke and Stuckenschmidt (2009) present two approaches to form diagnosis. The former called local diagnosis is defined as a minimal Hitting set formed of the less confidence weighted correspondences. The later called global optimal diagnosis is defined as the smallest diagnosis with respect to the total of confidences. Qi et al (2009) propose a conflict based mapping revision operator for the alignment revision. The operator is based on incision function that select from each minimal conflict set correspondences with less confidence values. These correspondences are then discarded from the alignment to establish coherency. The authors show that this operator can be characterized by two logical postulates adapted from some existing postulates for belief base revision: the relevance and consistency postulates (See chapter 2).

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

3.8 Conclusion

In this chapter we have reviewed the main approaches of the ontology evolution as well as the alignment evolution problems. Guiding by the fixed requirements of the problem of alignment evolution, we have discussed the applicability of ontology evolution frameworks to the problem of alignment evolution under ontology change. First, we have discussed globally how ontology evolution frameworks can embed these requirements in their change process. Then we have discussed their outcomes for every requirement. We conclude that these frameworks should be adapted in order to embed the alignment evolution problem. Besides, the alignment evolution should be separated from ontology evolution since alignment depending artifacts may create confusion with depending artifacts of ontologies.

We distinguished two classes of alignment evolution approaches: adaptive and perfective maintenance and corrective maintenance. The adaptive and perfective approaches modify the alignment according to the detected changes in ontologies. These approaches don’t consider explicitly the alignment consistency. Hence, no guaranties are given to product a consistent alignment after evolution. While the corrective maintenance approaches check and resolve inconsistences after change. The main challenge for these approaches is how to ensure a consistency alignment with a minimal of change. A promise investigation is to apply belief revision theory for alignment evolution problem. The work of Euzenat (2015) is a first step to understand revision in alignments of ontologies that may help to consider the problems of base revision.

Chapter 4.ontology change: Identification and Semantics on Alignment

4.1 Introduction

The change in ontologies may trigger change in depending artifacts such as other ontologies, instances, annotations, and alignments. In general, ontology evolution approaches don't study the evolution of all such depending artifacts. Instead, they describe general frameworks for the evolution of depending artifacts regardless of their natures. As we have seen in the discussion of the previous chapter, these approaches don't meet all the expected requirements for a system of alignment evolution problem. Furthermore, since alignment depending artifacts may create confusion with depending artifacts of ontologies the alignment evolution should be separated from ontology evolution. Following these observations, we give in this chapter an independent change process for the problem of alignment evolution under ontology change. First, we outline the general process then we instantiate the two first phases of this process, namely, the change identification phase and the semantics of change phase.

The objective of the change identification phase is detecting what has been changed in a version relatively to another and to make explicit this change in a machine readable format. Approaches of change detection and representation are ontology language dependent which hamper them to reach the wished consensus. In our approach (Zahaf, 2012; Zahaf and Malki, 2016a; 2016b), we consider a general format that can encompass any ontology language. Checking and resolution of alignment inconsistency are the main objective of the semantics of change phase. Different ways may exist for resolving the same inconsistency. One criterion that can guide the resolution is the minimal change. Sometimes, it is inevitable to sacrifice the minimal change against the consistency satisfaction. The challenge question is how to ensure the compromise between the consistency and the minimal of change constraints. To reach this objective, we have investigated

how to apply belief base revision theory for alignment evolution problem (Zahaf and Malki, 2016a).

The remainder of this chapter is organized as follows. In section 4.2, we give an independent change process for the problem of alignment evolution under ontology change. Section 4.3 outlines the models of ontologies and alignments considered in our framework. We instantiate the two first phases of the change process, namely, the change identification phase and the semantics of change phase in the sections 4.4 and 4.5 respectively. We conclude the chapter in section 4.6.

4.2 Alignment Evolution Process

As already mentioned in the introduction of this thesis, an alignment evolution under ontology change system should (1) facilitate the ontology change identification for maintainers, (2) evolve alignment from a consistent state to another consistent state, (3) conduct to a new consistent state with a minimal of change, and (4) permit to maintainers validating the new alignment by accepting the change, recovery from unnecessary changes, adapting the change, tracking it, or cancelling all the change. Hence, the alignment evolution is a process rather than a simple task that aims to keep the alignment consistent as much as possible with the updates of ontologies on which it depends on. To fulfill the above requirements, we propose the following alignment change process: a phase for the ontology change identification, a phase for the semantics of change, a phase for the change validation, and a phase for the change implementation. Figure 8 outlines this process.

Figure 8: The ontology alignment change process

(a) **Ontology change identification**: in open and distributed environments such as the semantic web where ontologies and alignments are submitted to different authorities, the journal of change is often available in an unreadable machine

format. Ontology evolution frameworks may not meet the ontology engineer's needs when he requests an ontology change. Instead, he turns to use ontology development tools which are not dedicated to the evolution of ontologies but to their modification. Unfortunately, the published new version is only associated with a list of change in human readable format which may hamper its exploitation by automatic tools. Even ontology evolution approaches deliver evolution logs that store the implemented change in machine readable format, maintainers of alignments may not share the same interpretation for the same change and they prefer to create their own set of change which might be different from the delivered set of changes. Maintainers want to identify and make explicit the ontology change in order to understand what happen and correctly update their alignments. The ontology change in this phase is obtained by comparing versions of the same ontology. Comparing versions aims to detect what has been changed in a version relatively to another and to make explicit this change in a machine readable format.

(b) **Semantics of change**: the objective of this phase is resolving alignment inconsistencies due to ontology change. As ontologies evolve from a consistent state to another, alignment evolution should follow this change by a transition to a new consistent state. Alignment consistency can be expressed as a set of constraints qualified as hard since their violation makes the alignment obsolete and useless. We distinguish three types of consistency for the problem of alignment evolution under ontology change. Alignment correspondences refer only to entities that belong to the aligned ontologies. The deletion of these ontological entities breaks the structure of the concerned correspondences. An alignment which has such correspondences is structurally inconsistent. An alignment should preserve its structure after the ontology change. We call such constraint, the structure preservation constraint. Ontologies are logical theories. Even, ontologies ensure their logical consistencies after the change; they can't preserve this consistency when they are used jointly with alignment. To preserve the logical consistency of ontologies, we should prevent the alignment from generating inconsistencies as logical consequences. We call such constraint, logical consistency preservation. Ontologies are the pillar of the semantic web; alignments maintainers may have not the permission to modify the changed

ontologies in order to establish the consistency of alignments. In other words, alignments maintainers should accept the ontological change and modifying alignments is the only possible way to establish the new consistency. Accepting the change may not be respected if some removed knowledge still entailed by alignments. In both cases, alignments should follow the ontology change by preserving it. We call such constraint, the ontological change preservation.

Many solutions can satisfy the above consistency constraints when we evolve the alignment. One of them is the empty alignment where we discard all its correspondences. It is obvious that empty alignment satisfies structure preservation since it doesn't connect any entities. Because we assumed that local ontologies are revised and bugs are fixed, no knowledge propagation is expected. Consequently, the empty alignment satisfies the constraints of consistence preservation and ontological change preservation. The empty alignment doesn't make any sense from a practical point of view and we need to compute the new alignment from scratch. An ideal solution is to change only the relevant correspondences that cause problems. We call such constraint, the constraint of minimal change. While the consistency constraints are qualified as hard we qualify the minimal change as a soft constraint. Since the violation of this constraint don't hamper the use of alignments.

(c) **Change validation**: alignment evolution is a knowledge intensive task which can't be fulfilled without the involvement of users. During the phase of semantics change, the system resolves the different types of inconsistencies by proposing changes on alignment. Proposed changes should be review by users before implementation. Alignments maintainers may validate the change, recover the unnecessary changes, adapt, track, or cancel the change by keeping connection with the old version of the evolved ontology if it is available. Hence, the objective of this phase is to rationalize and to facilitate the interaction between users and the system. To rationalize the interaction, the system should give explanations to inconsistencies and justify the proposed change as well. Even the objective of the previous phase is resolving inconsistencies with a minimal change; the proposed change may still containing unnecessary changes. Hence, the user can request the recovering of these changes. Detecting such changes is not an easy task. Thanks to inconsistency checking and explanation as well as change justification; the user

will adapt the change, reject it partially or reject it completely if justifications are not convincing. The system may propose discarding some correspondences while the user may choose to adapt them instead. The user adapts the change by adding, deleting, or modifying correspondences. He modifies correspondences by changing the type of their semantic relations, renaming entities, rates correspondences by attaching a new confidence values, or annotates them. All changes are stored on a journal of change which helps for change tracking.

(d) **Change implementation**: the change in previous phases has been done on a copy of the original old alignment. After the change is validated by users, the system confirms the change by implement it and delivers the new alignment and the final associated change. The final change is the difference between the old and the new delivered alignment. The format of both alignment and associated change should be machine readable. This allows the parsing and exploitation of changes by maintenance tools of depending applications.

This change process is general and can be implemented in different ways. In the rest of this chapter, we describe our approaches to concretize the two first phases: the phase of ontology change identification and the semantics of change phase. Before that, we present the models of ontologies and alignments used in our approaches.

4.3 Ontology and alignment models

4.3.1 Ontology Model

As it is stated in chapter 3, an ontology is an axiomatization of the intended meaning of a vocabulary used by a conceptualization of some area of interest. According to Uschold and Gruninger (2004), an ontology ranges from simple set of terms (folksonomy) 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, very significant knowledge). In this dissertation, we consider an ontology as a logical theory which consist of a set of axioms that specify the intend interpretation of a

vocabulary. Kalfoglou & Schorlemmer (2003) and Grimm & al (2011) represent ontologies as a pair (S, A), where S is a signature to designate a vocabulary and A is a set of the axioms. The signature of an ontology is the set $S = C \sqcup P \sqcup R \sqcup I$, 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. Axioms act as constraints for interpretations of this vocabulary. An interpretation which satisfies all axioms of an ontology constitutes a model of that ontology. Ontologies are expressed in logical languages such as RDF, RDFS and OWL. These languages provide a consequence relation between axioms of the language and ontologies.

Definition 4.1 (Ontology Consequence). An axiom δ is a logical consequence of an ontology O (noted $O \models \delta$) if and only if every model of O satisfies δ.

We denote by $Cn(O) = \{\delta | O \models \delta\}$ the closure set of logical consequences of an ontology O. We assume the logic of logical consequence relation satisfy the following properties:

Inclusion $O \subseteq Cn(O)$

Iteration $Cn(O) \subseteq Cn(Cn(O))$

Monotonicity if $O' \subseteq O$ then $Cn(O') \subseteq Cn(O)$

Compactness if $O \models \delta$ then, there is some subset $O' \subseteq O$ such that $O' \models \delta$.

Definition 4.2 (Inconsistent Ontology). An ontology O is inconsistent if and only if O has no model. Otherwise, it is consistent.

Usually, inconsistency checking is turned to contradictory axioms entailment checking (Hussain et al., 2011). When all models of an ontology lead to an unsatisfiable concept, we say that ontology is incoherent (Flouris et al., 2006). A concept is unsatisfiable if no individual belongs to that concept for all interpretations.

4.3.2 Alignment Model

Ontology matching is the task to detect links between elements from two ontologies. These links are referred to as correspondences and express semantic relations. According to Euzenat and Shvaiko (2013), we define a correspondence as follows and introduce an alignment as set of correspondences.

Definition 4.3 (Correspondence and Alignment). Given two ontologies o_1 and o_2, let Q be a function that defines sets of matchable elements $Q(o_1)$ and $Q(o_2)$. A correspondence between o_1 and o_2 is a 4-tuple (e, e', r, n) such that $e \in Q(o_1)$, $e' \in Q(o_2)$, r is a semantic relation, and $n \in [0; 1]$ is a confidence value. An alignment M between o_1 and o_2 is a set of correspondences between o_1 and o_2. We restrict r to be one of the semantic relations from the set $\{\text{Equivalence}(\equiv), \text{Subsomption}(\sqsubseteq), \text{Disjunction}(\perp)\}$.

In order to reason about alignment, we use a version of reductionist semantics (Meilicke & Stuckenschmidt, 2009) called natural semantics. 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 merged ontology.

Definition 4.4 (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 aligned ontology is defined by

$o_1 \cup_M o_2 = o_1 \cup o_2 \cup \text{trans}(M)$.

Example 4. The transformation of the alignment M of example 1 to axioms is as follows.

$$Trans(M) = \begin{Bmatrix} 1{:}\, PhD\ Student \equiv 2{:}\, PhD\ Student, \\ 1{:}\, Researcher \equiv 2{:}\, Researcher, \\ 1{:}\, Lecturer \equiv 2{:}\, Lecturer, \\ 1{:}\, Employee \equiv 2{:}\, Employee \end{Bmatrix}$$

Definition 4.5 (Alignment consequence): An axiom δ is a consequence of an alignment M between two ontologies o_1 and o_2 if and only if δ is a logical consequence of the aligned ontology $o_1 \cup_M o_2$. We denote this relation by $M \models \delta$.

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

Example 5. it is clear that $O_2 \nvDash Researcher \sqsubseteq Lecturer$ *but since,* $M \vDash 1{:}Researcher \equiv 2{:}Researcher$, $1{:}Researcher \sqsubseteq Lecturer$, $1{:}Lecturer \equiv 2{:}Lecturer$, *we can derive that* $M \vDash 2{:}Researcher \sqsubseteq Lecturer$. *Likewise, from* $M \vDash 1{:}Researcher \equiv 2{:}Researcher$, $1{:}Researcher \sqsubseteq Lecturer$, *we derive* $M \vDash 2{:}Researcher \sqsubseteq 1{:}Lecturer$. $2{:}Researcher \sqsubseteq 1{:}Lecturer$ *is an image of some correspondence between* $2{:}Researcher$ *and* $1{:}Lecturer$.

In the context of the natural semantics, we can easily verify that the alignment consequence relation satisfies the following properties:

Inclusion $M \subseteq Cn(M)$

Iteration $Cn(M) \subseteq Cn(Cn(M))$

Monotonicity if $M' \subseteq M$ then $Cn(M') \subseteq Cn(M)$

Compactness if $M \vDash \delta$ then, there is some subset $M' \subseteq M$ such that $M' \vDash \delta$.

Some alignment consequences are undesirables and can affect the consistency of ontologies or the whole aligned ontology. In this case, the alignment is inconsistent. When an ontology is inconsistent, the aligned ontology is also inconsistent. Since inconsistency is due to ontologies and not to alignment, we can't consider this case as an alignment inconsistency.

Definition 4.6 (Alignment Inconsistency): given an aligned ontology $o_1 \cup_M o_2$, M is inconsistent with respect to o_1 and o_2 if and only if both ontologies o_1 and o_2 are consistent but the aligned ontology $o_1 \cup_M o_2$ is inconsistent. Otherwise, M is consistent.

Example 6. $o_2^{"}$ *is inconsistent. Consequently, the aligned ontology* $o_1 \cup_M o_2^{"}$ *is also inconsistent. Since, inconsistency is due to the ontology* $o_2^{"}$ *and not due to the alignment M, we cannot consider M as inconsistent. However, if we consider* o_3 *which is consistent but since* $M \vDash Researcher \perp Lecturer$ *and* $M \vDash lecturer(Ahmed), Researcher(Ahmed)$, $o_1 \cup_M o_3$ *is inconsistent and hence, M is inconsistent.*

In other words, a consistent alignment preserves the consistency of ontologies. Alignment coherency which is a particular type of consistency ensures the satisfiability preservation of ontological entities by the alignment.

Definition 4.7 (Incoherent Alignment): given an aligned ontology $o_1 \cup_M o_2$, M is incoherent with respect to o_1 and o_2 if and only if both ontologies o_1 and o_2 are coherent but the aligned ontology $o_1 \cup_M o_2$ is incoherent. Otherwise, M is coherent.

Example 7. Following example 6, If we remove the assertion Phd Student (Ahmed) from o_3*, we get*

$M \models Researcher \perp Lecturer, PhD\ Student \sqsubseteq lecturer, PhD\ Student \sqsubseteq Researcher$ *. So, Phd Student becomes unsatisfiable and hence the alignment* M *is incoherent*

4.4 Ontology change identification

Ontology change is the result of any significant ontology modification. An ontology is an axiomatization of the intended meaning of a vocabulary used by a conceptualization of some area of interest. The modification can touch the meaning axiomatization of the vocabulary or the vocabulary itself. The vocabulary change is the set of added or deleted vocabulary elements. The axiomatic change is the set of added or deleted axioms. This leads to a simple ontology of change (See Figure 9). This format of change representation is general enough to encompass any ontology language.

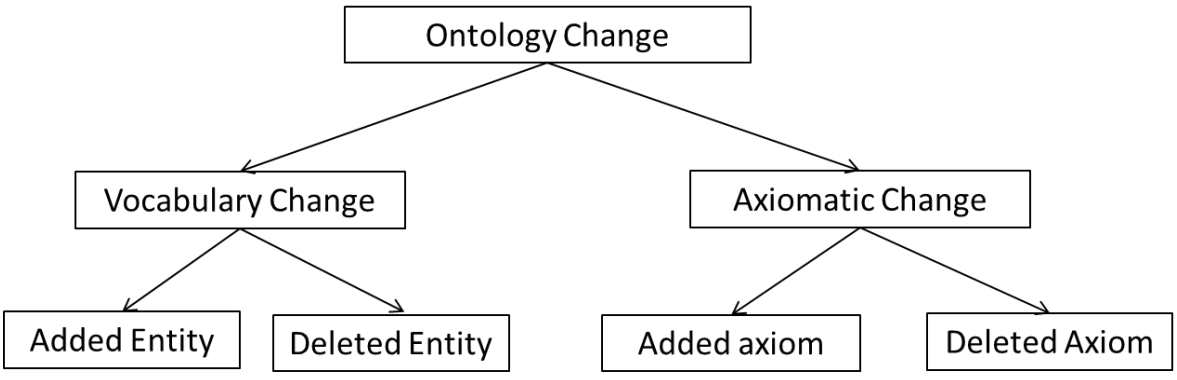

Figure 9: an ontology of change

For the purpose of the alignment between versions, we have developed a method (Zahaf, 2012) to compute the difference between versions. In this method, we consider the ontological change identification as the set theoretical difference between signatures and axioms. These operations use the output of version matching operation to compute persistent signatures and persistent axioms respectively. The set theoretical difference operation between the total signature and the persistent one constitutes the ontological change in signature. Similarly, the set theoretical difference operation between the set of axioms and the persistent one gives the ontological change in axioms. 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 version matching output. Finally, the obtained difference is refined by discarding axioms that are still entailed. Table 2 schematizes the described algorithm. We use the following notation: S_i^p to denote the set of persistent signature of a version i. S^- denotes the set of removed signature. S^+ is the set of added signature. Similarly, A_i^p is the set of persistent axioms of a version i. A^- is the set of deleted axioms and A^+ is the set of added axioms.

Example 8. the output of algorithm 1 for computing the difference between the two versions o_2 and o_3 is illustrated by the following sets.

$$S^- = \{Employee\}, \qquad A^- = \begin{Bmatrix} PhD\ Student \sqsubseteq Lecturer, \\ Lecturer \sqsubseteq Employee \end{Bmatrix}, \qquad S^+ = \emptyset,$$

$$A^+ = \{Researcher \perp Lecturer\}$$

4.5 Semantics of change

We consider the alignment evolution under ontology change as the set of changes on correspondences of alignment to fulfil the satisfaction of some semantics constraints. We distinguish three types of changes on alignments: expansion, contraction and revision. An expansion is a set-theoretically adding of correspondences to an alignment. It can happen following adding new ontological entities and we need to align them with others entities. A revision change restores

the alignment consistency following adding new correspondences or new axioms in ontologies. A contraction is to discard correspondences when concerned entities are deleted from ontologies or some successfully removed axioms from ontologies still logical consequences of the alignment. We consider in this framework, the revision change when new axioms in ontologies make alignment inconsistent and the contraction change when successfully removed axioms still a logical consequence of alignment.

Table 2 : ***Ontology change identification algorithm***

Algorithm 1: ontological change identification
ontologicalChange (H, o_2, o_3) Input: o_2, o_3 // two versions of the same ontology H // H is a mapping between o_2, o_3 Output : S^-, S^+ // set of removed vocabulary and the set of added vocabulary respectively A^-, A^+ // set of deleted axioms and the set of added axioms respectively 1. $S_1^p \leftarrow$ persistentSign(H, o_2) 2. $S_2^p \leftarrow$ persistentSign(H, o_3) 3. $S^- \leftarrow S_1 - S_1^p$ 4. $S^+ \leftarrow S_2 - S_2^p$; 5. $A_1^p \leftarrow$ persistentAxioms(H, o_2) 6. $A_1^p \leftarrow$ persistentAxioms(H, o_3) 7. $A^- \leftarrow A_1 - A_1^p$ 8. $A^+ \leftarrow A_2 - A_2^p$; 9. for $\delta \in A^-$ do 10. if $o_3 \vDash H(\delta)$ 11. then $A^- \leftarrow A^- - \{\delta\}$; 12. for $\delta \in A^+$ do 13. if $o_2 \vDash H^-(\delta)$ // H^- is the inverse of H 14. then $A^+ \leftarrow A^+ - \{\delta\}$ Return $\{(S^-, A^-), (S^+, A^+)\}$

Viewing ontologies as logical theories allows us to consider the aligned ontology formed by the alignment and the connected ontologies as a logical theory too. In practical, ontologies are encoded in knowledge bases managed by knowledge systems to have access to and to reason about domain knowledge (Grimm & al., 2011). The set of axioms contained in these bases constitutes the explicit knowledge and implicit knowledges are logical consequences of them. Hence, our approach follows the belief base revision approach instead of the AGM model. More precisely, our objective is to adapt the kernel contraction framework (Hansson, 1994) to design rational operators for the alignment evolution under ontology change. In what follows, we present two operators for the alignment evolution under ontology change. The former discards correspondences when some successfully removed axioms from ontologies still logical consequences of the alignment violating the constraint of change preservation. We call it the alignment kernel contraction. The latter baptized the alignment kernel consolidation which restores the logical consistency of the alignment following adding new axioms in ontologies.

4.5.1.1 Alignment Kernel Contraction

Given an alignment M between two ontologies o_1 and o_2 and α is a successfully removed axiom from one ontology, the outcome of a contraction is to compute a subset of M that fails to imply α. The alignment kernel contraction consists in finding the set of minimal subsets of M that imply α. We call this set, the kernel of M by α and we denote it by $M \Vdash \alpha$. We adapt the base kernel definition (See Definition 2.6) to define the alignment kernel as follows:

Definition 4.8 (Alignment Kernel): the kernel of M by α ($M \Vdash \alpha$) is the set of elements M' such that:

$$\begin{cases} M' \sqsubseteq M \quad (\text{subset of M}) \\ M' \vDash \alpha \quad (\alpha \text{ is a consequence of M}) \\ \forall M'' \sqsubseteq M', M'' \nvDash \alpha \ (\text{and it is minimal}) \end{cases}$$

We call an element of the kernel ($M \Vdash \alpha$) an α-Alignment kernel.

Example 9. Considering example 3, alignment M between o_1 and o_3 violates the change preservation and still entails the contracted axiom $\alpha = PhD\ Student \sqsubseteq Lecturer$. The kernel of M by α is as follows:

$$K = \left\{ \begin{Bmatrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: Lecturer =_{0.62} 2: Lecturer, \end{Bmatrix}, \begin{Bmatrix} 1: Lecturer =_{0.62} 2: Lecturer, \\ 1: PhD\ Student =_{1.00} 2: PhD\ Student \end{Bmatrix} \right\}$$

Lemma 4.1: the following two conditions are equivalents.

For all $M' \sqsubseteq M$, $M' \models \alpha$ iff $M' \models \beta$

$M \parallel \alpha = M \parallel \beta$

Proof: (necessary condition). We should demonstrate that $M \parallel \alpha \subset M \parallel \beta$ and $M \parallel \beta \subset M \parallel \alpha$ follows from the first condition. We give a proof for the first inclusion and the same proof holds for the second inclusion.

Let $M' \in M \parallel \alpha$, we should demonstrate that $M' \in M \parallel \beta$. From the alignment kernel definition (See definition 4.8), $M' \in M \parallel \alpha$ means $M' \sqsubseteq M$, $M' \models \alpha$ and $\forall M'' \sqsubseteq M', M'' \not\models \alpha$. According to the first condition, we have $M' \models \beta$. From the alignment kernel definition (See definition 4.8), we conclude that $M' \in M \parallel \beta$.

(Sufficient condition). We should demonstrate that the first condition, for all $M' \sqsubseteq M$, $M' \models \alpha$ iff $M' \models \beta$ follows from the second $M \parallel \alpha = M \parallel \beta$. Let $M' \models \alpha$ for $M' \subset M$. By alignment compactness, there exists a subset $M'' \subset M'$such that $M'' \models \alpha$. Let M'' be the minimal one. From the alignment kernel definition (See definition 4.8), $M'' \in M \parallel \alpha$. According to the second condition, we have $M'' \in M \parallel \beta$ and hence, $M'' \models \beta$. By alignment monotony, we conclude that $M' \models \beta$. The same proof holds for the inverse.

The alignment kernel contraction uses a function to discard from M at least one correspondence from each α-Alignment kernel. We call such function an alignment incision function. We adapt the base incision function definition (See Definition 2.7) to define an alignment incision function as follows:

Definition 4.9 (Alignment Incision function): an incision function σ for M is a function that for all α :

$$\begin{cases} \sigma(M \parallel \alpha) \sqsubseteq \bigsqcup(M \parallel \alpha) \\ \text{if } \emptyset \neq X \in M \parallel \alpha, \text{then } X \cap \sigma(M \parallel \alpha) \neq \emptyset \end{cases}$$

Example 10. A possible incision function for the kernel K of the example 9 is.

$$I = \begin{Bmatrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: PhD\ Student =_{1.00} 2: PhD\ Student \end{Bmatrix}$$

Therefore, we define the alignment kernel contraction by adapting the base kernel contraction definition (See definition 2.8) as follows:

Definition 4.10 (Alignment Kernel Contraction): let M be an alignment between two ontologies o_1 and o_2, α is a successfully removed axiom from one ontology, and σ is an alignment incision function, the alignment kernel contraction of M by α is the operator defined as: $M -_{\sigma} \alpha = M \setminus \sigma(M \parallel \alpha)$

Example 11. If we consider the incision function of example 10, the kernel contraction of M by PhD Student ⊑ Lecturer is,
$M -_{\sigma} \alpha = \{1: Lecturer =_{0.62} 2: Lecturer\}$

We adapt the postulates success, inclusion, core-retainment and uniformity of the belief base kernel contraction (See Theorem 2.3) to define the postulates that the alignment kernel contraction should satisfy. For the alignment kernel contraction, a success means that successfully removed axioms from ontologies should not be regenerated again by the alignment after contraction. The postulate of success corresponds to the ontology change preservation constraint in case of removing knowledge. Satisfying the core-retainment means a correspondence is discarded only if it is responsible somehow for implying the contracted axiom. We consider the postulate of core-retainment as a criterion to ensure the minimal change requirement. Furthermore, the alignment kernel contraction satisfies two others postulates. The postulate of inclusion ensures no new correspondence should be added to the alignment when realizing contraction. The uniformity postulate expresses there is no reason that the contraction by two different but logically related axioms is not the same. These two postulates are out of the fixed requirements for the alignment evolution problem. However, we need them to more characterizing the contraction change. The postulates of inclusion together with success ensure the pure contraction[10] and the uniformity postulate ensures a deterministic change. The following theorem represents the alignment kernel contraction operator.

[10] According to Hansson (1999), "in pure contraction a belief should be given up without being replaced by any new belief".

Theorem 4.1 (Alignment Kernel Contraction Representation theorem): An operator $-$ is an alignment kernel contraction of an alignment M between two ontologies o_1 and o_2 for a successfully removed axiom α from an ontology if and only if it satisfies the following postulates:

[success] if $\nvDash \alpha$ then $M - \alpha \nvDash \alpha$

[Inclusion] $M - \alpha \subseteq M$

[Core − retainment] if $c \in M$ and $c \notin M - \alpha$, then there is a subset M' of M such that, $M' \nvDash \alpha$ but $M' \sqcup \{c\} \vDash \alpha$

[uniformity] if it hollds for all $M' \subseteq M$ that $M' \vDash \alpha$ if and only if $M' \vDash \beta$, then $M - \alpha = M - \beta$

Proof: (necessary condition). Let $-_\sigma$ be an alignment contraction operator such that $M-_\sigma\alpha = M \setminus \sigma(M \parallel \alpha)$ for some incision function σ and demonstrates that it satisfies the postulates: success, inclusion, core-retainment, and uniformity.

Success and inclusion follow directly from the operator definition. Suppose $c \in M$ and $c \notin M-_\sigma\alpha$, then $c \in \sigma(M \parallel \alpha)$. From the definition of the alignment incision function (See definition 4.9), we have $\sigma(M \parallel \alpha) \sqsubseteq \bigsqcup(M \parallel \alpha)$, so there is some set A such that $c \in A \in (M \parallel \alpha)$. Let $M' = A \setminus \{c\}$, then $M' \nvDash \alpha$ but $M' \sqcup \{c\} \vDash \alpha$. This satisfies core-retainment. From lemma 4.1, For all subset M' of M, $M' \vDash \alpha$ if and only if $M' \vDash \beta$ is equivalent to $M \parallel \alpha = M \parallel \beta$. Since σ is a function then $\sigma(M \parallel \alpha) = \sigma(M \parallel \beta)$. It follows $M \setminus \sigma(M \parallel \alpha) = M \setminus \sigma(M \parallel \beta)$. Hence, $M-_\sigma\alpha = M-_\sigma\beta$. We conclude that $-_\sigma$satisfies success, inclusion, core-retainment, and uniformity.

(Sufficient condition). Let – be a contraction operator on an alignment M such that the four postulates are satisfied. We are going to demonstrate that – is a kernel contraction. For that purpose, let σ be such that for α: $\sigma(M \parallel \alpha) = M \setminus M - \alpha$. We need to verify that σ is an incision function for M. To be that, it must: first, be a function and second such that it satisfies i) $\sigma(M \parallel \alpha) \sqsubseteq \bigsqcup(M \parallel \alpha)$ and ii)if $\emptyset \neq X \in M \parallel \alpha$, then $X \cap \sigma(M \parallel \alpha) \neq \emptyset$. Furthermore, we need to verify that – applied to M coincides with $-_\sigma$.

Proof that σ is a function. Let α and β be two correspondences such that $M \parallel \alpha = M \parallel \beta$. It follows from Lemma 4.1 and uniformity that $M - \alpha = M - \beta$. Following our definition $\sigma(M \parallel \alpha) = M \setminus M - \alpha$, we conclude $\sigma(M \parallel \alpha) = \sigma(M \parallel \beta)$.

Proof that the condition i) $\sigma(M \parallel \alpha) \sqsubseteq \bigsqcup(M \parallel \alpha)$ is satisfied. Let $c \in \sigma(M \parallel \alpha)$. By core-retainment, it follows that there is some $A \sqsubseteq M$ such that $A \not\models \alpha$ and $A \sqcup \{c\} \models \alpha$. By compactness, there is some finite subset $A' \sqsubseteq A$ such that $A' \sqcup \{c\} \models \alpha$. From $A' \not\models \alpha$ and $A' \sqcup \{c\} \models \alpha$, it follows that there is some α-alignment kernel A'' that contains c. Then, $c \in A'' \in \bigsqcup(M \parallel \alpha)$.

Proof that the condition ii) $\text{if } \emptyset \neq X \in M \parallel \alpha, \text{then } X \cap \sigma(M \parallel \alpha) \neq \emptyset$ is satisfied. Suppose that $\emptyset \neq X \in M \parallel \alpha$. Then $\not\models \alpha$ and by success, we have $M - \alpha \not\models \alpha$. Since $X \models \alpha$, we may conclude that $X \not\subset M - \alpha$, i.e., that there is some c such that $c \in X$ and $c \notin M - \alpha$. Since $X \sqsubseteq M$, it follows $c \in M \setminus M - \alpha$; i.e., $c \in \sigma(M \parallel \alpha)$. Thus, $c \in X \cap \sigma(M \parallel \alpha)$ which is sufficient to show condition ii) satisfaction.

Proof that – applied to M coincides with $-_\sigma$. By inclusion ($M - \alpha \subseteq M$) and our definition of $\sigma(M \parallel \alpha) = M \setminus M - \alpha$, it follows $M - \alpha = M \setminus \sigma(M \parallel \alpha)$. This finishes the proof.

4.5.1.2 Alignment Kernel consolidation

We define an alignment consolidation as all operation that makes consistent an alignment. An alignment kernel consolidation is the alignment kernel contraction by the contradictory axiom (i.e., ⊥ (a)). For each inconsistency element of the alignment kernel, the consolidation removes from the alignment at least one element that is responsible for this inconsistency. Formally,

Example 12. The alignment M *between the ontologies* o_1 *and* o_3 *of example 3 is inconsistent. The kernel consolidation of* M *is.*

$$K = \left\{ \begin{Bmatrix} 1{:}\,Researcher =_{0.62} 2{:}\,Researcher, \\ 1{:}\,Lecturer =_{0.62} 2{:}\,Lecturer, \end{Bmatrix}, \begin{Bmatrix} 1{:}\,Lecturer =_{0.62} 2{:}\,Lecturer, \\ 1{:}\,PhD\ Student =_{1.00} 2{:}\,PhD\ Student \end{Bmatrix} \right\}$$

We can define the alignment kernel consolidation by adapting the base kernel consolidation operator (See definition 2.10) as follows:

Definition 4.11 (Alignment Kernel Consolidation): let M be an alignment between two ontologies o_1 and o_2 and σ an alignment incision function, the alignment kernel consolidation of M is the operator defined as:

$M!_{C,\sigma} = M \setminus \sigma(M \Vdash\!\!\perp (a))$

Example 13. If we choose $I = \{1{:}\,Lecturer =_{0.62} 2{:}\,Lecturer\}$ *as an incision function for the kernel of example 11 the kernel consolidation of the alignment M is* $M!_{C,\sigma} = \begin{Bmatrix} 1{:}\,PhD\ Student =_{1.00} 2{:}\,PhD\ Student, \\ 1{:}\,Researcher =_{0.62} 2{:}\,Researcher \end{Bmatrix}$

We consider also, an operator that makes an alignment coherent as a particular alignment consolidation. In this case, the contraction is done by the following subsumption axiom (i.e., $C \subset \perp$).

By adapting the postulates characterizing the belief base kernel consolidation operator (See theorem 2.5) the alignment kernel consolidation is characterized by the following postulates: consistency, inclusion and core-retainment. The postulate of consistency corresponds to the logical consistency preservation constraint. Again, the postulate of core-retainment expresses the minimal change criterion. The postulate of inclusion imposes contraction as the only possible strategy to restore consistency. By construction, the alignment kernel consolidation operator doesn't modify the ontologies but only alignment is modified to restore consistency. Hence, the operator satisfies the change preservation constraint. The following theorem represents the alignment kernel consolidation operator.

Theorem 4.2 (Alignment Kernel Consolidation Representation theorem): An operator ! is an alignment kernel consolidation of an alignment M between two ontologies o_1 and o_2 if and only if satisfy the following postulates:

[Consistency] $M! \nVdash\!\!\perp (a)$

[Inclusion] $M! \subseteq M$

[Core − retainment] if $c \in M$ and $c \notin M!$, then there is a subset M' of M such that, $M' \nVdash\!\!\perp (a)$ but $M' \sqcup \{c\} \Vdash\!\!\perp (a)$

The same holds for alignment consolidation in case of a coherent alignment. However, we rename the consistency postulate by coherency postulate.

Proof: (necessary condition). Let $M!_{C,\sigma} = M \setminus \sigma(M \parallel\!\perp (a))$ be an alignment kernel consolidation operator. Inclusion follows from the definition. We are going to demonstrate that $M!_{C,\sigma}$ satisfies the consistency. Suppose $\perp (a) \in M!_{C,\sigma}$, by compactness there is some subset $A \subseteq M!_{C,\sigma}$ and $A \vDash \perp (a)$. Hence, there is some inclusion-minimal subset $C \subseteq A$ such that $C \vDash \perp (a)$. Thus, $C \in (M \parallel\!\perp (a))$. Due to $\nvDash \perp (a)$, then $C \neq \emptyset$. Following the definition of the alignment incision function $C \cap \sigma(M \parallel\!\perp (a)) \neq \emptyset$. Then, there is some element $c \in C$ and $c \in \sigma(M \parallel\!\perp (a))$. According to the operator definition, $c \in C \subseteq A \subseteq M!_{C,\sigma}$. But $c \in \sigma(M \parallel\!\perp (a))$. Hence, C cannot be a subset of $M!_{C,\sigma}$. We conclude that $\perp (a) \notin M!_{C,\sigma}$. Let $c \in M$ and $c \notin M!_{C,\sigma}$. Then,$c \in \sigma(M \parallel\!\perp (a)$. Following the definition of the alignment incision function, there is some set C such that $c \in C \in (M \parallel\!\perp (a))$. Let $X = C \setminus \{c\}$. Then, $X \nvDash \perp (a)$ and $X \sqcup \{c\} \vDash \perp (a)$. This demonstrates the satisfaction of the core-retainment postulate.

(Sufficient condition). Let ! be an alignment operator on M such that the three postulates of inclusion, consistency, and core-retainment are satisfied. We are going to demonstrate that ! is an alignment kernel consolidation based on some function σ. For that purpose, let σ be such that for α: $\sigma(M \parallel \alpha) = M \setminus M!$. We need to verify that σ is an incision function for M and to verify that the operator ! applied to M coincides with $!_{C,\sigma}$. Be that, it must satisfying i) $\sigma(M \parallel\!\perp (a)) \sqsubseteq \bigsqcup(M \parallel\!\perp (a))$ and ii) if $\emptyset \neq A \in M \parallel\!\perp (a)$, then $A \cap \sigma(M \parallel\!\perp (a)) \neq \emptyset$.

Clearly σ is a function. To show the first condition i) $\sigma(M \parallel\!\perp (a)) \sqsubseteq \bigsqcup(M \parallel\!\perp (a))$, let $c \in \sigma(M \parallel\!\perp (a))$. It follows from the core-retainment that there is some A such that $A \subseteq M$, $A \nvDash \perp (a)$ and $A \sqcup \{c\} \vDash \perp (a)$. By compactness, there is some subset $A' \subseteq A$ such that $A' \sqcup \{c\} \vDash \perp (a)$. Let A'' an inclusion-minimal subset of A' such that $A'' \sqcup \{c\} \vDash \perp (a)$. Hence, there is some α-Alignment kernel C such that $c \in C \in (M \parallel\!\perp (a))$. For the second condition ii), let $\emptyset \neq A \in M \parallel\!\perp (a)$. By consistency, $M! \nvDash \perp (a)$. Since $A \vDash \perp (a)$, by monotony $A \nsubseteq M!$. That there is a correspondence $c \in A$ and $c \notin M!$. Since, $A \subseteq M$, $c \in M \setminus M!$. This means $c \in \sigma(M \parallel\!\perp (a))$. Thus, $c \in A \cap \sigma(M \parallel\!\perp (a))$. This finishes the proof that σ is an alignment incision function.

It follows from the inclusion and our definition of σ (i.e., $\sigma(M \parallel \alpha) = M \setminus M!$), $M! = M \setminus \sigma(M \parallel \alpha)$. We conclude that M! is an alignment kernel consolidation (i.e., $M! = M!_{C,\sigma}$).

4.5.1.3 Confidence based operations

Incision as general functions are extra-logical means to choose between correspondences of an alignment. We define a special incision function based on confidence values associated to correspondences. For that purpose, we introduce an order relation on correspondences of the alignment. This relation uses confidence values associated to correspondences to establish such order. The correspondence in an α-Alignment kernel that has the less confidence value constitutes an element of this function.

Definition 4.12 (Confidence based incision function): Given, $(M \parallel \alpha)$ an alignment kernel M with respect to an axiom α, σ_c is a confidence based incision function for M if and only if for all α:

$$\begin{cases} \text{(i) } \sigma_c(M \parallel \alpha) \sqsubseteq \cup (M \parallel \alpha) \\ \text{(ii) if } \emptyset \neq X \in (M \parallel \alpha), \text{then } X \cap \sigma_c(M \parallel \alpha) \neq \emptyset \\ \text{(iii) if } c = (e, e', r, n) \in \sigma_c(M \parallel \alpha), \text{then there exists } C \in (M \parallel \alpha) \text{ such that,} \\ c \in C \text{ and } n = \min\{n_i | (e_i, e'_i, r_i, n_i) \in C\} \end{cases}$$

The conditions say that a confidence based incision function takes a correspondence from each α-Alignment kernel and this correspondence should have the less confidence value when compared with the others.

Example 14. Since the correspondence $1{:}\,Lecturer =_{0.62} 2{:}\,Lecturer$ *presents the less confidence value relatively to others, the confidence based incision function for the kernel K of example 9 is* $I = \{1{:}\,Lecturer =_{0.62} 2{:}\,Lecturer,\}$

Therefore, we can define the confidence based alignment kernel contraction as follows:

Definition 4.13 (Confidence based Alignment Kernel Contraction): let M an alignment between two ontologies o_1 and o_2, α a successfully removed axiom from one ontology and σ a confidence based alignment incision function, the confidence

based alignment kernel contraction of M by α is the operator defined as: $M-\sigma_c\alpha = M \setminus \sigma_c(M \parallel \alpha)$

Similarly, we define the confidence based alignment kernel consolidation,

Definition 4.14 (Confidence based Alignment Kernel Consolidation): let M be an alignment between two ontologies o_1 and o_2 and σ_c a confidence based alignment incision function, the confidence based alignment kernel consolidation of M is the operator defined as:

$$M!_{C,\sigma_c} = M \setminus \sigma_c(M \parallel\!\perp (a))$$

4.6 Conclusion

In this chapter, we have presented a framework for the problem of the alignment evolution under ontology change. Briefly, we have outlined the different phases of the change process of the alignment evolution. Then, we have instantiated the two first phases of this process, namely, the change identification phase and the semantics of change phase. In this framework, we have considered an ontology as a vocabulary and a set of axioms specifying the meaning of this vocabulary. Following this model of ontologies, an ontology change is the set of added and deleted vocabulary elements on the one hand and the set of added and deleted axioms on the other hand. This format of change representation is general enough to encompass any ontology language. Encoding of ontologies as knowledges bases within knowledge systems allowed us to consider the belief base revision theory for designing rational operators for the alignment evolution problem. More precisely, we have adapted the kernel framework of the base revision theory to design two general operators: the kernel contraction and the kernel consolidation for the alignment evolution. These operators are characterized by a set of postulates. Some of these postulates match the constraints of alignment consistency and the minimal change. Based on confidence values associated to the alignment correspondences, we have given two particular operators: the confidence based kernel contraction and the confidence based kernel consolidation.

Besides, the kernel framework can support the alignment maintainer with means for change explanation and justification. Indeed, an alignment kernel is a set of minimal subsets responsible of alignment inconsistency. This is exactly what a justification is in debugging of ontologies. Furthermore, incision functions select among these justifications the accused correspondences to establish consistency. The notions of kernel and incision function play an important role to rationalize the interaction between maintainers and the alignment evolution system.

Chapter 5. Methods

5.1 Introduction

The framework of the previous chapter describes general operators for the resolution of the alignment inconsistency. Theses operators base their actions on the notions of the alignment kernel and the incision function. The alignment kernel is the set of the minimal subsets of correspondences causing the violation of the alignment consistency. The incision function selects from each element of the kernel at least one correspondence for resolving inconsistencies. In this chapter, we will see how to compute the alignment kernel as well as the corresponding incision functions. Incision functions are the Hitting set of the alignment kernel since it intersect each element of this kernel. Hence, we adapt the Hitting set algorithm (See Section 2.3) of the diagnosis theory to compute the kernel and all the corresponding incision functions. We give another algorithm to compute the confidence based incision functions as well. Both algorithms have an exponential time in the worst case. To reduce the complexity, we sacrifice the computing of all confidence based incision functions by computing only one incision function in an efficient time. The new algorithm runs in logarithmic time at worst.

The defined operators deal only with inconsistencies of logical and change preservation types. In this chapter, we extend our framework by a global method (Zahaf and Malki, 2016b) that deals with all types of consistency, namely, the logical consistency, the change preservation consistency, and the structural consistency as well. This method is an orchestration of a set of operations each of which is designed to take care of one aspect of the alignment change process. The remainder of this chapter is organized as follows. Section 5.2 presents the proposed algorithms for computing the alignment kernel and their corresponding incision functions as well. Section 5.3 presents the proposed algorithms for computing the confidence based incision functions. In section 5.4, we present and

discuss the strength and the weakness of our proposed global method. We conclude the chapter in section 5.5.

5.2 Computing alignment Kernel and Incision Functions

The algorithm to find an α-Alignment kernel is an adaptation of the algorithm presented in (Baader et al., 2007) to compute a minimal subset of an ontology that is responsible for an entailment of a given subsumption axiom (see Table3). It consists in removing each element of M and testing if the resulting alignment still implies the axiom α. If this is not the case the element is reintroduced in M. The result of this process is a set $M' \sqsubseteq M$ that do imply α which is minimal. Similar to the algorithm presented in (Baader et al., 2007), algorithm 2 can compute an α-Alignment kernel in polynomial time in the size of the aligned ontology.

Table 3: α-Alignment kernel algorithm.

Algorithm 2: α-Alignment kernel
α-Alignment kernel (M, o_1, o_2, α) Input : o_1, o_2 // two ontologies M // M is an alignment between o_1 and o_2 α // α is an axiom Output : M // an α-Alignment kernel 1. for $c \in M$ 2. do 3. if $M \setminus \{c\} \models \alpha$ 4. then $M \leftarrow M \setminus \{c\}$ 5. return M

*Example 15. Following example 9, we demonstrate how to compute the first α-*Alignment kernel *by using the algorithm 2. Let α be PhD Student ⊑ Lecturer.*

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

2. Checks $M \setminus \{1: PhD\ Student =_{1.0} 2: PhD\ Student\} \models \alpha$ *(line 3). So it removes* $1: PhD\ Student =_{1.0} 2: PhD\ Student$ *from M (line 4).*

3. Checks $M \setminus \{1{:}\,Researcher =_{0.62} 2{:}\,Researcher\} \not\models \alpha$. *Then it does not change M (line 3).*

4. Checks $M \setminus \{1{:}\,Lecturer =_{0.62} 2{:}\,Lecturer\} \not\models \alpha$, *so it does not change M(line 3).*

5. Return $M = \begin{Bmatrix} 1{:}\,Researcher =_{0.62} 2{:}\,Researcher, \\ 1{:}\,Lecturer =_{0.62} 2{:}\,Lecturer \end{Bmatrix}$ *which is an* α-Alignment kernel *(line 5).*

Table 4: Alignment kernel and Incision functions algorithm

Algorithm 3: Alignment Kernel and Incision functions
AlignmentKernelAndIncisionFct $(\mathrm{M}, \mathrm{o}_1, \mathrm{o}_2, \alpha)$ Input : $\mathrm{o}_1, \mathrm{o}_2$ // two ontologies M // M is an alignment between o_1 and o_2 α // α is an axiom Output : AKernel // an Alignment kernel Incision // set of incision functions 1. Incision ← ∅ 2. Stack ← Empty 3. C ← α-Alignment kernel $(\mathrm{M}, \mathrm{o}_1, \mathrm{o}_2, \alpha)$ 4. AKernel ← $\{C\}$ 5. for c ∈ C 6. do insert $\{c\}$ in the top of the stack 7. While Stack not Empty 8. do Hn ← last element of the stack 9. remove last element of the stack 10. If $M \setminus \{Hn\} \models \alpha$ 11. Then C ← α-Alignment kernel $(M \setminus \{Hn\}, \mathrm{o}_1, \mathrm{o}_2, \alpha)$ 12. AKernel ← AKernel ∪ $\{C\}$ 13. for c ∈ C 14. do insert $Hn \cup \{c\}$ in the top of the stack 15. Else Incision ← Incision ∪ $\{Hn\}$ 16. End.

To compute the alignment Kernel and incision functions, we adapt the Hitting set algorithm proposed by Reiter (1987) to diagnose systems (See Chapter 2). The alignment kernel is the collection of all α-Alignment kernel. By definition (See definition 4.9), the alignment incision function intersects each α-Alignment kernel. Hence, it seems naturel to consider the incision function as a Hitting set (See definition 2.14) of the alignment kernel. The nodes of the tree are labeled by α-Alignment kernels and edges are labeled by the elements of these α-Alignment kernels. However, the kernel is not given explicitly and we should compute it. At each node, an α-Alignment kernel of the set M \ H(n) is computed if such an α-Alignment kernel exists. Otherwise, H(n) is an alignment incision function. Unfortunately, the Hitting set algorithm has an exponential time (Rymon, 1991). Table 4 outlines this algorithm. The progress of the algorithm is illustrated by the example 16 and figure 10 as well.

Example 16. Following the example 15, we want to compute incision functions.

1. *Algorithm 3 starts by computing one α-Alignment kernel. Let it the same as in example 15:*

   $C = \left\{ \begin{matrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: Lecturer =_{0.62} 2: Lecturer, \end{matrix} \right\}$ *(line 3-4)*

2. *Push* $\{c\}$ *into the stack for every element of C (line 5-6). The content of the stack is stack* $= \left\{ \begin{matrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: Lecturer =_{0.62} 2: Lecturer \end{matrix} \right\}$.

3. *Get the last element of the stack into Hn.*

   $Hn = \{1: Lecturer =_{0.62} 2: Lecturer\}$ *and*

   $stack = \{1: Researcher =_{0.62} 2: Researcher\}$ *(line 8-9).*

4. *Checks* $M \setminus \{1: Lecturer =_{0.62} 2: Lecturer\} \nvDash \alpha$, *then* $Incision = \{\{1: Lecturer =_{0.62} 2: Lecturer\}\}$ *(lines 10 and 15).*

5. *Loop line(7).*

6. *Get the last element of the stack into Hn.*

   $Hn = \{1: Researcher =_{0.62} 2: Researcher\}$ *and* $stack = \emptyset$ *(line 8-9).*

7. *Checks* $M \setminus \{1: Researcher =_{0.62} 2: Researcher\} \vDash \alpha$ *line(10), then*

8. *Run algorithm 2 again, we obtain*

   $C = \left\{ \begin{matrix} 1: PhD\ Student =_{1.00} 2: PhD\ Student, \\ 1: Lecturer =_{0.62} 2: Lecturer, \end{matrix} \right\}$ *(line 11)*

9. Push $Hn \cup \{c\}$ into the stack for every element of C (line 13-14). The stack contains now,

$$stack = \left\{ \begin{Bmatrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: PhD\ Student =_{1.00} 2: PhD\ Student \end{Bmatrix}, \begin{Bmatrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: Lecturer =_{0.62} 2: Lecturer \end{Bmatrix} \right\}.$$

10. Loop line(7).

11. Get the last element of the stack into Hn.

$$Hn = \begin{Bmatrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: Lecturer =_{0.62} 2: Lecturer \end{Bmatrix} \text{ and stack}$$

$$= \left\{ \begin{Bmatrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: PhD\ Student =_{1.00} 2: PhD\ Student \end{Bmatrix} \right\} \text{ (line 8-9).}$$

12. Checks that $M \setminus \{Hn\} \not\models \alpha$ line(10), then

$$Incision = \left\{ \begin{matrix} \{1: Lecturer =_{0.62} 2: Lecturer\}, \\ \begin{Bmatrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: Lecturer =_{0.62} 2: Lecturer \end{Bmatrix} \end{matrix} \right\} \text{ (lines 10 and 15).}$$

13. Loop line(7).

14. Get the last element of the stack into Hn.

$$Hn = \begin{Bmatrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: PhD\ Student =_{1.00} 2: PhD\ Student \end{Bmatrix} \text{and stack} = \emptyset \text{ (line 8-9).}$$

15. Checks that$M \setminus \{Hn\} \not\models \alpha$ line(10), then Incision =

$$\left\{ \begin{matrix} \{1: Lecturer =_{0.62} 2: Lecturer\}, \\ \begin{Bmatrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: Lecturer =_{0.62} 2: Lecturer \end{Bmatrix}, \\ \begin{Bmatrix} 1: Researcher =_{0.62} 2: Researcher, \\ 1: PhD\ Student =_{1.00} 2: PhD\ Student \end{Bmatrix} \end{matrix} \right\}$$

16. The stack is empty, line(7).

17. End (line 16)

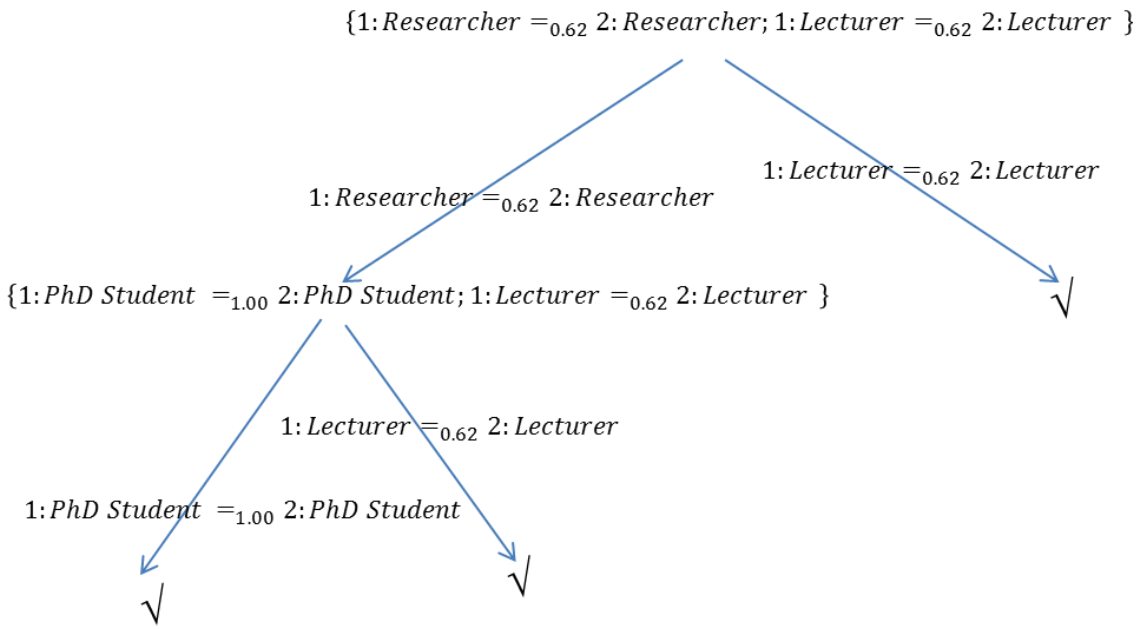

Figure 10: Hitting set tree of incision functions

5.3 Computing Confidence based incision functions

Confidence based incision functions select from each α-Alignment kernel the correspondence that has the less confidence value. We refine algorithm 3 by introducing a function that pick these correspondences before computing incision functions. Table 5 schematizes the refined algorithm. Figure 11 illustrates the outcome of this algorithm regarding the example 16.

Algorithm 4 reduces enormously the complexity time depending on confidence values attached to correspondences. If all correspondences present the same confidence value or no values the algorithm 4 is reduced to the algorithm 3. Both algorithms compute all incisions functions. In real applications, we sacrifice this need by computing only one incision function in efficient time. For that purpose, we adapt algorithm 4 to compute only one incision function. The algorithm 5 (See Table 6) acts as the binary search algorithm[11]. It develops just one branch of the tree which corresponds to the correspondence with lowest confidence value in the generated α-Alignment kernel. Hence, this algorithm runs in logarithmic time at worst.

[11] 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 5: Confidence based Incision functions algorithm

Algorithm 4: Confidence based Incision functions
ConfidenceBasedIncisionFct $(\mathrm{M}, \mathrm{o}_1, \mathrm{o}_2, \alpha)$ Input : $\mathrm{o}_1, \mathrm{o}_2$ // two ontologies M // M is an alignment between o_1 and o_2 α // α is an axiom Output : AKernel // an Alignment kernel Incision // set of incision functions 1. Incision ← $\emptyset$ 2. Stack ← Empty 3. C ← α-Alignment kernel $(\mathrm{M}, \mathrm{o}_1, \mathrm{o}_2, \alpha)$ 4. AKernel ← $\{C\}$ 5. minC ← CorrespWithLowestConfidValue(C) 6. for c ∈ minC 7. do insert $\{c\}$ in the top of the stack 8. While Stack not Empty 9. do Hn ← last element of the stack 10. remove last element of the stack 11. If $M \setminus \{Hn\} \models \alpha$ 12. Then C ← α-Alignment kernel $(M \setminus \{Hn\}, \mathrm{o}_1, \mathrm{o}_2, \alpha)$ 13. AKernel ← AKernel ∪ $\{C\}$ 14. minC ← CorrespWithLowestConfidValue(C) 15. for c ∈ minC 16. do insert $Hn \cup \{c\}$ in the top of the stack 17. Else Incision ← Incision ∪ $\{Hn\}$ 18. Return Incision 19. End.

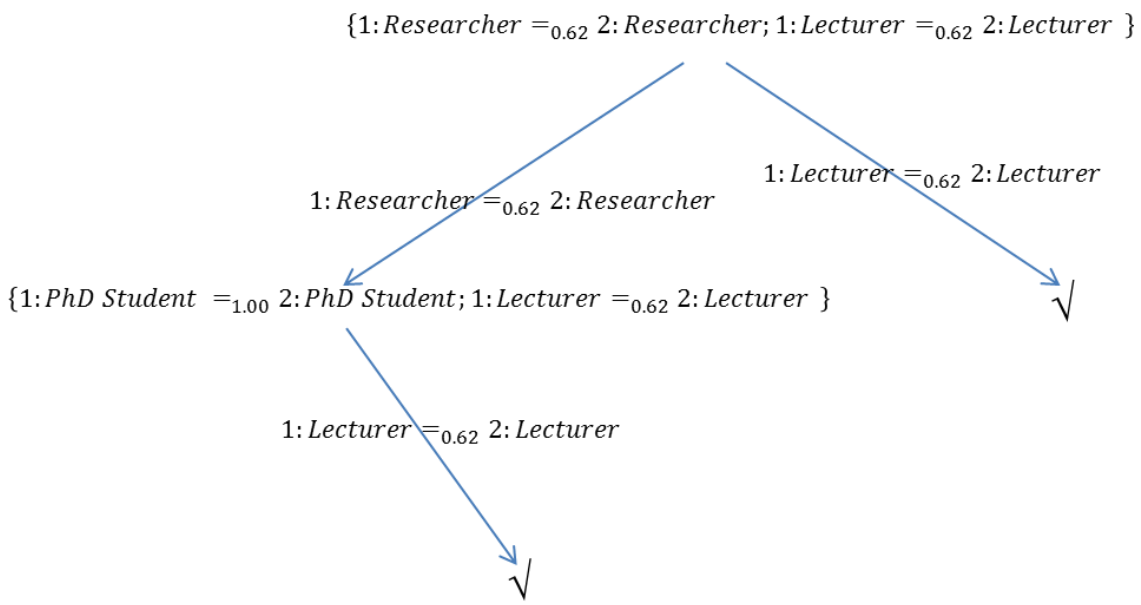

Figure 11: Hitting set tree of confidence based incision functions

Table 6: Binary search based incision function algorithm.

Algorithm 5: Binary search based incision function
BinarySearchBasedIncisionFunction $(\mathrm{M}, \mathrm{o}_1, \mathrm{o}_2, \alpha)$ Input : $\mathrm{o}_1, \mathrm{o}_2$ // two ontologies M // M is an alignment between o_1 and o_2 α // α is an axiom Output : $\mathrm{Incision}_c$ // an incision function 1. $\mathrm{Incision}_c \leftarrow \emptyset$ 2. **If** ($\mathbf{o_1} \nvDash \alpha$ *and* $\mathrm{o}_2 \nvDash \alpha$) 3. **while** $M \vDash \alpha$ 4. **do** 5. C $\leftarrow \alpha$-Alignment kernel $(\mathrm{M}, \mathrm{o}_1, \mathrm{o}_2, \alpha)$ 6. Clv $\leftarrow$ CorrespWithLowestConfidValue(C) 7. $\mathrm{Incision}_c \leftarrow \mathrm{Incision}_c \cup$ {Clv} 8. $M \leftarrow M \setminus$ {Clv} 9. **Return** $\mathrm{Incision}_c$

5.4 Alignment evolution method

5.4.1 Overview

The alignment evolution under ontology change is a process that needs an orchestration of operations in order to resolve the problem. Figure 12 schematizes the orchestration of these operations.

Given an alignment M between two ontologies o_1 and o_2, o_3 is another version of the second ontology, the method computes an alignment between versions of the changed ontology following a matching operation. The ontological change operation serves of this alignment to compute the ontological change as the difference between the two versions of the evolved ontology. The proposed method uses confidence based operations as main operations[12] for the alignment revision. It embeds these two operations in a generic one that adapts its input according to the type of change. This operation sets its input to removed axioms in case of an alignment contraction or to contradictory axioms in case of an alignment consolidation. The objective of this operation is to identify the correspondences that cause violation of the logical consistency and the ontology change preservation constraints and to give means to choose among them which must be eliminated.

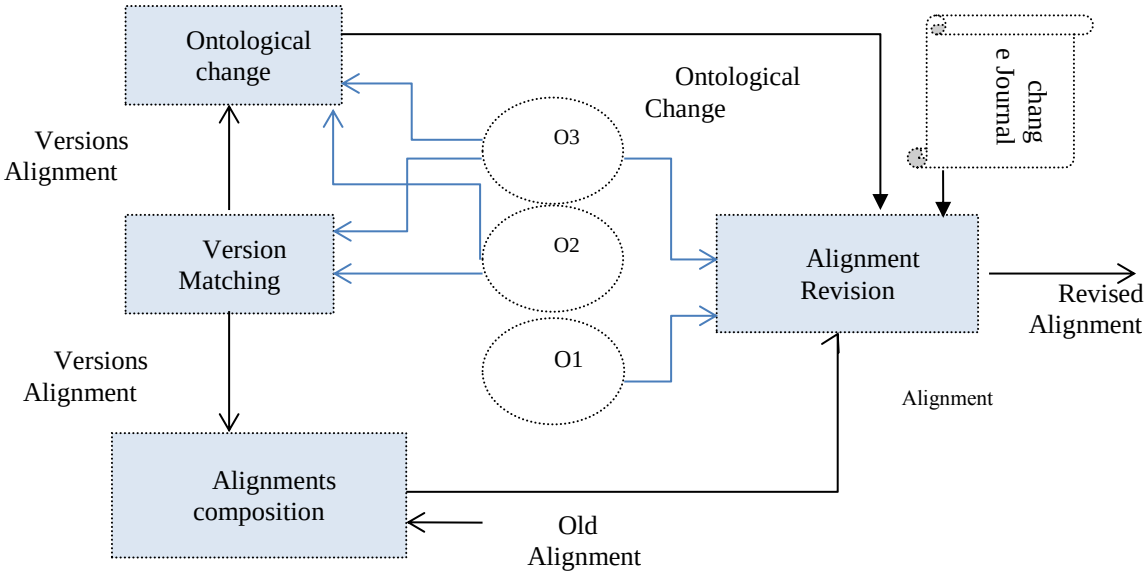

Figure 12: Alignment Evolution Method

[12] A confidence based operation uses a confidence based incision function to select correspondences from the α-alignment kernel sets. In (Zahaf and Malki, 2016b), we have used the name alignment diagnosis instead the name of incision function and the name of minimal conflict set instead the name of α-alignment kernel.

Confidence based operators discard correspondences from the old alignment based on confidence values associated to these correspondences. However, these confidence values represent trust degrees in correspondences before the ontology change. The challenging question is how to update these trust degrees in order to reflect the change. For that purpose and to deal with the violation of structure preservation constraint, we give a third operation which acts as a pre-operator of the others to fulfil the global objective. This operation gives new confidence values to correspondences after the change on which the others operations base their decisions. To do that, this operation composes the old alignment with some computed alignment between versions of the changed ontology following a matching operation. The composition forms a new alignment whose correspondences are established between elements of the new version of the changed ontology and elements of the other ontology. The correspondences keep their semantic relations while they update their associated confidence values by taking the average number between the old values and the computed ones between versions.

In what follows we discuss the outcomes of these operations on the minimal change requirement and the alignment evolution constraints satisfaction.

5.4.2 Discussion

Version matching: version matching is the starter operation of the evolution process. It serves of any matcher as a plugin to computes an alignment between versions of the evolved ontology. The alignment expresses equivalence relations between matched entities in both versions. The ontological change and the alignments composition operation base their actions on the alignment produced by the version matching operation. Hence, the correctness and completeness of the version matching operation are determining factor for the final result of the global method. Indeed, missing correspondences in the produced alignment are missed in the final alignment as well. In addition, erroneous correspondences may make the operation of revision more burdensome by unnecessary inconsistencies. This can have drastic results on the minimal change requirement. Instead to remove only the concerned correspondences, some others are removed not by relevance to the problem but following gaps in the versions matching operation. For these reasons, we recommend the careful examination of versions matching results.

Ontological change: As illustrated by algorithm 1, this operation computes the ontological change as the set-theoretical difference between the signatures of ontology versions as well as between the sets of axioms. The correctness and completeness of this ontological change operation depend on the operation of version matching. More the versions matching is precise and complete, more the ontological change is correct and complete.

Alignments composition: the alignment composition operation noted by ∘ composes between the old alignment M and the alignment H obtained following the version matching operation. M ∘ H is given by the set :

$$\mathrm{M} \circ \mathrm{H} = \left\{ (e_1, e_3, r_1 \circ r_2, avg(n_1, n_2)) \;\middle|\; \exists e_2 \in o_2, \; (e_1, e_2, r_1, n_1) \in M \wedge (e_2, e_3, r_2, n_2) \in H \right\}$$

Where $r_1 \circ r_2$ is defined by table 7 which illustrates the composition of basic relations. avg(n_1, n_2) is a function that return the average values of confidence values n_1 and n_2.

Table 7 : Composition of basic set-theoretical relations

∘	=	>	<	∩	⊥
=	=	>	<	∩	⊥

By definition, the alignment composition eliminates systematically the correspondences connecting entities that are deleted after ontology change. These correspondences are responsible for the violation of the structure preservation constraint. By construction, the alignment between versions contains only equivalence relations. Consequently, no change affects the semantic relations of remained correspondences as it is illustrated in table 7. Similar to the ontological change operation, the correctness and completeness of the composition operation depend on the correctness and completeness of the version matching operation. If the versions matching operation computes erroneous correspondences, the alignment composition also generates erroneous ones which may gravely complicate the problem. Furthermore, if the operation misses some

correspondences, the completeness of the alignment composition decreases. Here, we also would like to stress that the operation of version matching should be conducted under carefully eyes.

Alignment revision operation: it is a generic operation that adapts its input according to the type of change. For removed axioms, the method calls the confidence based alignment kernel contraction operation. In case of the alignment inconsistency, the method calls the confidence based alignment kernel consolidation operation. In order to be a full automatic method, these operations embed the algorithm 5 of the binary search based incision function which computes one incision function for a given alignment. While the performances of the previous operations depend only on techniques used to achieve version matching, the performances of the alignment revision depend on the underlying representation languages of ontologies. The correctness of this operation needs ontology languages to be monotones and compact. Fortunately, like OWL such languages exist. The alignment natural semantics respects 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 constraints of consistency and the ontological change preservation. Like the previous operations, the minimal change is also at stake for the alignment revision. Confidence based incision functions may discard more correspondences than necessary. This is can happen since some correspondences may have the same confidence value within an α-Alignment kernel.

Example 17. we assume that algorithm 5 performs from left to right. Following example 15, the outcome of the algorithm 5 is $\left\{\begin{matrix} 1{:}\,Researcher =_{0.62} 2{:}\,Researcher, \\ 1{:}\,Lecturer =_{0.62} 2{:}\,Lecturer \end{matrix}\right\}$. *Instead of selecting only one correspondence, algorithm 5 selects both correspondences at once.*

Unfortunately, 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.

Besides, the binary search based incision function is biased by the arrangement of the correspondences in the alignment. The outcome of algorithm 5 depends on the arrangement of correspondences in the alignment.

Example 18. If we assume that the correspondence $1{:}\,\text{Lecturer} =_{0.52} 2{:}\,\text{Lecturer}$ *has 0.72 instead of 0.62 as a confidence value, the outcome of the algorithm 5 is illustrated by Figure 13. If we assume that algorithm 5 performs from left to right, the arrows with solid line in figure 13 show the outcome of this algorithm. Figure 13 shows that instead of selecting only the correspondence* $1{:}\,Lecturer =_{0.72} 2{:}\,Lecturer$*, the algorithm 5 also selects the correspondence* $1{:}\,Researcher =_{0.62} 2{:}\,Researcher$ *as well. This is because the correspondence* $1{:}\,Researcher =_{0.62} 2{:}\,Researcher$ *is arranged before* $1{:}\,Lecturer =_{0.72} 2{:}\,Lecturer$ *in the alignment.*

However, we have no means to know how to arrange the correspondences in such a way to get a minimal incision function.

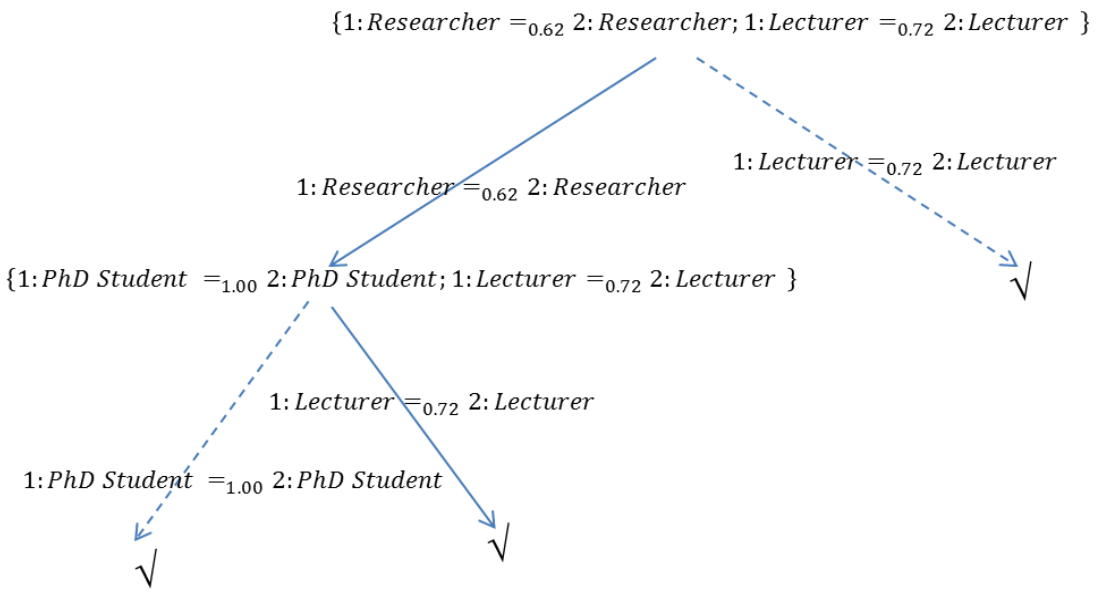

Figure 13: Binary search based incision functions

5.5 Conclusion

In this chapter, we have presented the computational aspect of our framework. We have adapted the Hitting set algorithm of the diagnosis theory to compute the alignment kernel as well as the corresponding incision functions. As a result, we have proposed different algorithms with different complexities varying from an exponential to a polynomial time.

Besides, this chapter extends the proposed framework by a global method that deals with all types of consistency, namely, the logical consistency, the change preservation consistency, and the structural consistency as well. This method is an orchestration of a set of operations each of which is designed to take care of one aspect of the alignment change process. Finally, we have discussed the weakness and the strength of this method relatively to the minimal change requirement and the alignment consistency constraints satisfaction. This method takes the version matching at the starter operation of the whole process. Method's results are biased by the weakness and strength of this operation. Namely, incorrect version matching may make the resolution of inconsistency more burdensome by unnecessary inconsistencies. Furthermore, incomplete version matching may leads to missing of correspondences in the final result. This may have drastic results on the minimal change requirement

Chapter 6. Implementation and applications

6.1 Introduction

In chapter 4, we have described a formal evolution framework to support and guide maintainers of ontology alignments. The framework describes different operators for the resolution of the alignment inconsistency. Chapter 5 presents the computational aspect of this framework. Different methods with different complexities varying from exponential to polynomial time have been presented to concretize the defined operators. In this chapter, we focus on tools and their applications. We discuss a prototype implementation that serve as a proof of concept for the feasibility of the main ideas presented in this dissertation. Also, we discuss the experiences from applying some methods of our framework for demonstrating the limits of some approaches from the category of the adaptive and perfective alignment maintenance approaches. Mainly, these approaches rely on ontology matching techniques for evolving alignments. By selecting these methods we want to show neither ontology matching nor alignment debugging methods fit well for the alignment evolution problem. Hence, we signal the emergency need of dedicated methods to deal with the alignment evolution problem. The remainder of this chapter is composed of three sections. Section 6.2 discusses the implementation of the proposed architecture of our alignment evolution system. Section 6.3 discusses the advantage of our approach relatively to others in the category of the adaptive and perfective alignment maintenance approaches. We conclude the chapter in section 6.4.

6.2 Implementation

We have implemented a prototype of our alignment evolution system in java. The platform of this prototype is based on OWL API (Horridge & Bechhofer, 2011) for manipulating OWL ontologies and Align API (David et al., 2011) for manipulating alignments between them. The platform integrates pellet (Sirin et al.,

2007) as the main reasoning engine on OWL ontologies. In what follows we describe in a nutshell these APIs then we give the architecture of the alignment evolution system illustrating its different components on top of these APIs.

6.2.1 OWL API

The OWL API provides a collection of interfaces supporting the use of OWL ontologies within applications (Horridge & Bechhofer, 2011). The model explicitly supports the recent OWL 2 Recommendation[13]. The OWL API supports reading and writing ontologies in several syntaxes, including RDF/XML, Turtle, OWL/XML, OWL Functional Syntax, The Manchester OWL Syntax, KRSS Syntax[14] and the OBO flat file format[15]. Besides, the API allows the imports closure of ontologies written in different syntaxes.

An OWL ontology in the OWL API model is a set of OWL axioms (see Figure 14). The OWLOntology interface provides a point for accessing the axioms contained in an ontology. The OWLOntologyManager provides methods for creating, loading, changing and saving ontologies, which are instances of the OWLOntology interface.

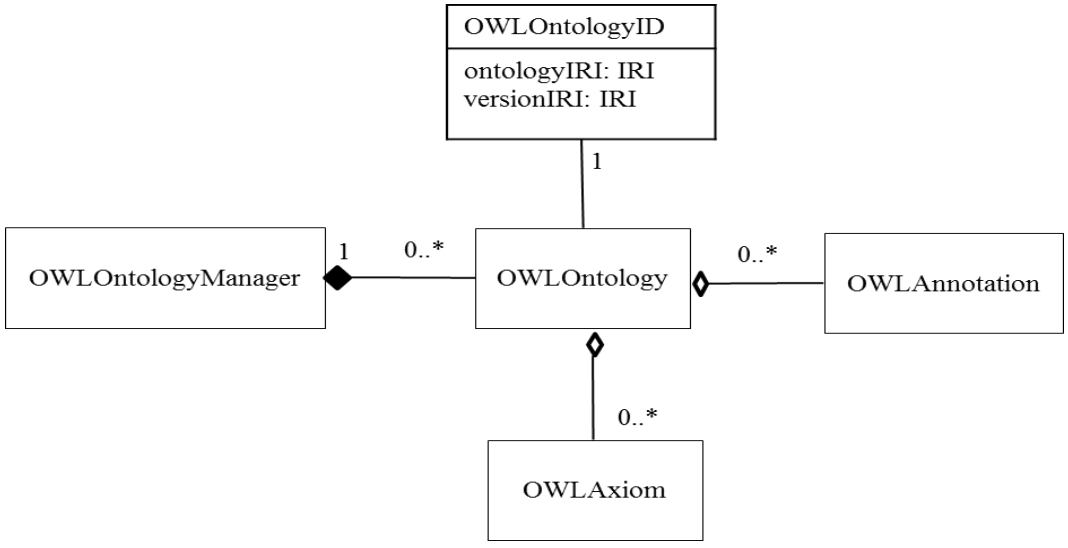

Figure 14: A UML diagram showing the management of ontologies in the OWL API.

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

Furthermore, the OWL API defines common interfaces for supporting tasks such as consistency checking, computation of class or property hierarchies and axioms entailment.

6.2.2 Alignment API

The Alignment API provides definitions of a set of Java interfaces and basic implementations of them (David et al., 2011). The main representational classes are represented in Figure 15. These classes provide methods for manipulating alignments such as adding correspondences to alignments and cutting correspondences under a confidence threshold.

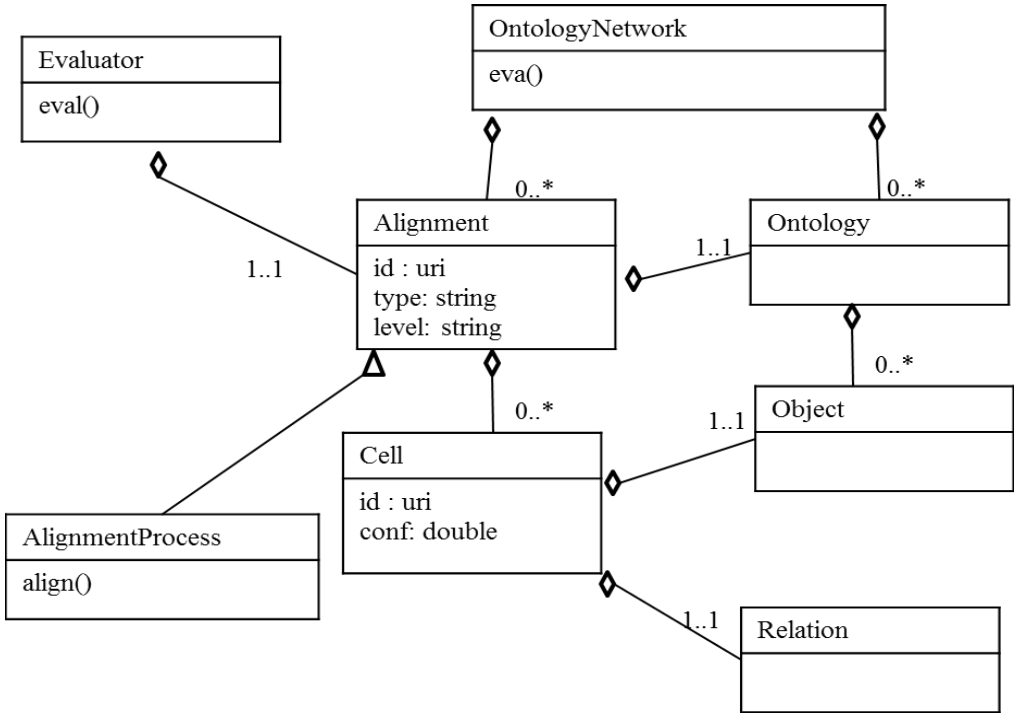

Figure 15: A UML diagram showing the management of alignments in the Alignment API.

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.

Furthermore, 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 the reference and a second alignment.

6.2.3 Architecture

The alignment evolution system embeds the OWL API and Alignment API libraries as a baseline for loading ontologies and for loading, modifying, and storing alignments. The different components of the system and the interaction between them are represented in Figure 16. An arrow interconnecting two components indicates a using relation between them. The ontology change component is responsible for identifying and representing the ontology change. This component implements the algorithm 1 (See chapter 4) for computing the difference between two versions of the same ontology. It performs version matching by invoking the Alignment API services. For the moment, we don't implement any specific matcher but we are satisfied by loading any alignment computed by any matcher between the concerned versions. The loaded alignment serves as a mapping for relating the same entities in both versions. We serve of OWL API libraries to represent and store the deduced change as instances of the ontology of change (See Figure 9).

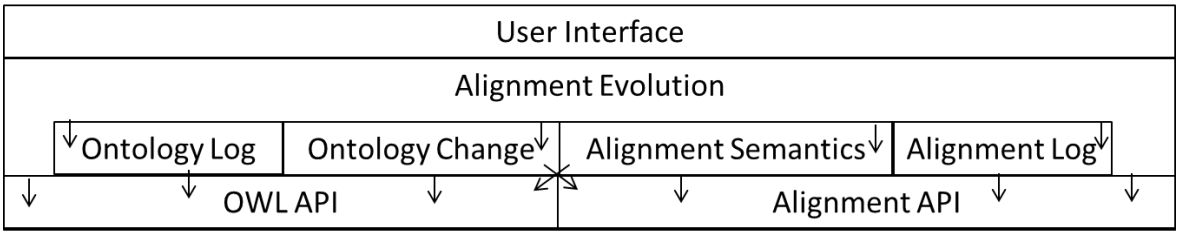

Figure 16: The architecture of the alignment evolution system.

Basically, the alignment evolution component implements the different operators for the problem of alignment evolution under ontology change. Namely, it implements the general operators for the alignment contraction and consolidation and the special operators of the confidence based alignment contraction and confidence based alignment consolidation. For that purpose, it

implements algorithms 2, 3, 4, and 5 for computing an α-Alignment kernel, the alignment kernel and incision functions, confidence based incision functions, and binary search based incision function respectively. Besides, this component implements the alignment evolution method (See chapter 5). The implemented methods of this component invoke the service for computing the ontology change of the ontology change component and they use reasoning services offers by the alignment semantics component. Within the alignment semantics component, we can check the alignment consistency, the entailment of an ontology axiom or an alignment correspondence, and deduce new alignments by the composition of related alignments which have a common ontology. The separation of reasoning services from alignment evolution methods may enhance the scalability to integer different alignment semantics. Actually, our system embeds the natural semantics where an alignment is converted to a global ontology by using the Alignment API libraries. The global ontology consists of importing the aligned ontologies and the conversion of the alignment correspondences to a set of axioms. Consequently, reasoning on alignment turns to reasoning on this global ontology. We serve of OWL API services for connecting to any reasoning engine. By default, our system integrates pellet (Sirin et al., 2007). The alignment change is entrusted to the alignment log component for representing, storing, and tracking the alignment change.

6.3 Applications

There are many ways to evaluate our proposed methods. For instance, it is judicious to evaluate the performances of the proposed method for alignments evolving vis-à-vis the change minimal principle since it satisfies the core-retainment which is only a weak form of it. Furthermore, we compare these performances with those of alignment evolution approaches. But, it seems priority to investigate if all these approaches deal with the identified alignment evolution constraints. For this purpose, we applicate the algorithm 5 of our framework (See chapter 5) to demonstrate the limits of some approaches from the category of the adaptive and perfective maintenance approaches (See chapter 3). We define measures for measuring the accuracies of these approaches as well as the cost needed to change them. We compare the obtained results by these methods in the

context of alignment evolution with those that can be obtained in ontology matching context highlighting the divergence between both contexts. Hence, we signal the emergency need of dedicated methods to deal with the alignment evolution problem. However, some challenges such as the data set on which to perform tests, the accuracy, and the cost of change measures used to evaluate the results hinder the completion of our objectives.

6.3.1 Selected evolution methods.

Basically, the adaptive and perfective maintenance approaches rely on ontology matching techniques to evolve alignments. Some of them (Khattak et al, 2015) recomputed the affected correspondences from scratch by matching them with elements of the new version of the evolved ontology. Others (Groß et al, 2013) compose the result of matching between versions of the evolved ontology with the old alignment. In this experiment, we don't study these approaches as they are. Instead, we prepare different variants of them by diversifying the ontology matching tools used for generating the alignment. This strategy helps us avoiding any bias in favor of a particular ontology matching tool.

The ontology alignment evaluation initiative (OAEI[16]) organizes an annual workshop. The workshop knows the participation of a plethora of 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. In addition to matching, YAM++ (Ngo & Bellahsene, 2012), Lily (Wang & Xu, 2008), and ASMOV (Jean-Mary et al., 2009) integrate components for the alignment debugging (See chapter 3). YAM++[17] and Lily[18] are open software and they are available to download from the web. Even ASMOV[19] is commercial software and no shareware is available; its outputs for the systematic benchmark test are available on its web sites. For these reasons, we have selected these tools to consider three methods for representing methods of evolved alignments from scratch. Only YAM++ and Lily are used to generate the alignments between versions. Hence, we consider two methods for representing alignments

[16] http://oaei.ontologymatching.org/
[17] http://www.lirmm.fr/yam-plus-plus/
[18] http://cse.seu.edu.cn/people/pwang/lily.htm
[19] http://www.infotechsoft.com/products/asmov.aspx

composition methods. The semantic relations of the new correspondences are adapted according to table 7 (See chapter 5) and their new confidence values are the average of confidence values of old correspondences and those values of correspondences between versions. In total, we select five methods for this experiment. By selecting these methods we want to show neither ontology matching nor alignment debugging methods fit well for the alignment evolution problem.

6.3.2 The Data set.

The data set should contain a set of ontologies, alignments between these ontologies, and the ontology change as an explicit journal or as some versions of the changed ontologies. Regarding the ontologies, we have adapted a subset of the systematic benchmark test owned by OAEI[20], which is a coordinated international initiative that organizes the evaluation of ontology matching systems. The benchmark is formed from a bibliographic ontology, some of its variants ontologies and alignments between these variants. These alignments are used as references ones in order to measure the accuracy of ontology matching tools. The domain of this ontology is the bibliographic references. It is based on a subjective view of what must be a bibliographic ontology. The ontologies are described in OWL-DL and serialized in the RDF/XML format. The reference ontology is that of the test #101. It contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals.

In what concerns the ontology change, we chose the ontologies 103,104,203,223,230, and 233 then we formed the following three tests: # 101-103-104, #101-203-223, and #101-230-233. We consider the ontology 104 as a version of the ontology 103, the ontology 223 as a version of 203 and the ontology 233 as a version of 230. These selected versions are variants of the reference ontology 101 by discarding or adding some features. The version 103 is language generalization of 101. 104 is language restriction variant of 101. The version 203 is 101 without comments. While 223 is an expanded hierarchy of 101 and 233 is

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

devoid of hierarchy and properties. The version 230 is the flattened classes of 101. Hence, the formed data set covers the most possible changes between ontologies.

We also consider the alignments between the following pairs of ontologies: 101-103, 101-203, and 101-230 as the old alignments. Since ontology matching problem is not the same as alignment evolution problem, we cannot consider the alignments between the following pairs of ontologies: 101-104, 101-223, and 101-233 as new alignments for the considered old ones. Instead, we follow an alternative approach to measure the accuracy of methods without the need of these references alignments. Figure 17 schematizes this data set.

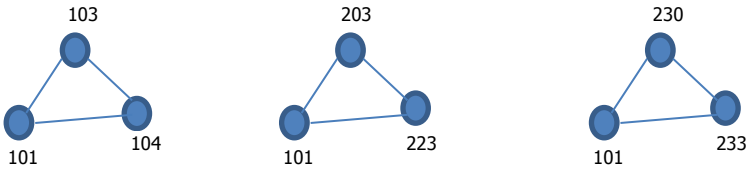

Figure 17: Data Set

6.3.3 Accuracy measures.

To show the limits of the selected methods when dealing with alignment evolution problem, we use the number of constraints violation by changed axioms. To compute the cost of change needed by these methods for evolving their outputs, we apply the cardinality based degree of incoherence measure proposed in (Meilicke & Stuckenschmidt, 2008). This measure gives an upper bound for precision without any knowledge of the reference alignment also known as the gold standard alignment. The cardinality based degree of incoherence C of an alignment M between two ontologies o_1 and o_2 is defined by $C(o_1, o_2, M) = \frac{|\Delta|}{|M|}$, where Δ is the minimal diagnosis of M relatively to the number of correspondences. The coherence degree of a method is given by $1 - C$ which is an upper bound for methods precisions. The cardinality based degree of incoherence C of an alignment M also gives the percentage of the number of correspondences that should be changed to establish consistency. Hence, this measure gives the cost of change needed to evolve the alignment. However, our algorithm 5 doesn't computing a minimal diagnosis relatively to the cardinality but an incision

function which cannot be always the minimal one. Nevertheless, applying our algorithm for computing this measure expresses at worst the cost of change for evolving an alignment. Consequently, the coherence degree measure gives the safe reusing percentage of an alignment after the ontology change. To measure performances of the selected methods in ontology matching context, we use the precision measure which expresses their correctness relatively to the available reference alignments.

6.3.4 Experimentation process.

The experimental process had conducted in two steps. In the first step, we generate the ontological change. In the second step, we use our algorithm 5 to show performances and limits of the selected alignment evolution methods to avoid the violation of the alignment evolution constraints. We compute the cost needed for evolving alignments. We serve of the cost result to compute the safe reusing percentage of alignments. We conclude the experimentation by comparing the accuracy of selected methods in both contexts of ontology matching and alignment evolution problem. In what follows, we detail these steps.

Table 8 : The ontology change of the data set.

Difference / Versions	**Deleted Signature**	**Added Signature**	**Deleted Axioms**	**Refined Deleted Axioms**	**Added Axioms**	**Refined Added Axioms**
103-104	0	0	11	11	0	0
203-223	0	34	49	9	78	78
230-233	52	8	220	220	0	0

Step 1 (Ontological change generation). To generate the ontological change, we use the version 2012 of the system YAM++ available on its website[21] for matching versions and we use our algorithm 1 (See chapter section 4) for computing the difference between versions. YAM++ prompts users to select matchers and to integrate instances, similarity propagation, or semantic verification in the

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

matching process. See (Ngo & Bellahsene, 2012) for more detail. Basically, the performance of our method for the ontological change generation relies on the persistent signature detection. For this reason and since there is no change in naming convention of signature elements for the selected versions, we have selected only the levenshtein[22] matcher. Table 8 summarizes the difference between each pair of versions. Axioms removed from 103 compared to 104 are domains for object and data properties. Besides the addition of new entities and related axioms to version 223, definitions of other entities had changed by adding axioms. The same holds for definitions of some entities in version 203 by removing axioms. The removing axioms are domains, ranges and some restrictions on properties. The comparison between versions 230 and 233 shows an addition of some entities and a removal of some entities and some axioms as well. The deletion of axioms is due to the removal of the hierarchy between entities.

Step 2 (methods performances and limitations). The objective of this step is to show the limit of the selected methods to avoid the violation of the alignment evolution constraints. Also we compare their performances in both contexts of ontology matching and alignment evolution problems. The experimentation had split into two folds: alignments composition methods and evolving alignments from scratch. In order to be fair, reference alignments are considered as old alignments and methods of the former generate new alignments by composition between them and the generated ones between versions. Besides selecting all proposed matchers, similarity propagation, and semantic verification are also integrated in YAM based matching process between versions. With regards to LILY based methods, we use its version 2 that is available for downloading on its website[23]. Lily presents a user friendly interface to configure some parameters. We choose 15 as the size of semantic subgraph and we enabled similarity propagation option. Since we deal with semantics properties of alignments in this step, these parameters setting are more than necessary in order to fit both systems with their full potentialities. In what concerns the latter, YAM, Lily, and ASMOV are tools used to compute the new alignments from scratch. We keep the same configuration of YAM and Lily as in the previous fold to generate alignments between 101-104,

[22] http://www.levenshtein.net/
[23] http://cse.seu.edu.cn/people/pwang/lily.htm

101-223 and 101-233. ASMOV presents outputs alignments between these ontologies on its website[24] and they are available for downloading.

In order to reach the drawn objective, we count the number of violations of the ontological change and consistency preservation constraints caused by the ontology change. When we applied algorithm 5 on the new computed alignments, we observed no consistency preservation constraint violation. But all of them violated the ontological change preservation constraint. Table 9 presents the average of results of this experiment. The first column designates the method. The second column shows the average size of old alignments used in the test. The third column shows the average size of the computed alignments between versions. The fourth column shows the average size of the new alignments generated by the selected methods. The fifth column shows the average number of constraints' violations.

At first glance, the number of constraints violation seems to be the same for all methods. However, we cannot confirm that all methods register the same score when dealing with this problem. Alignment quality depends on its content and its size. For instance, an empty alignment avoids completely the violation of constraints but it doesn't present any interest. The sixth column shows the average of diagnosis sizes that are sufficient to ensure constraints preservation. The seventh column expresses at worst the cost of change for evolving an alignment.

Table 9 : Limits of ontology matching tools and alignment debugging methods

Method	#Old	#Ver	#New	**#Violation**	#Delta	**Cost (%)**	**Coh° (%)**
YAM based Composition	89	73	73	**8**	9	**20**	**80**
Lily based Composition	89	71	71	**6**	6	**15**	**85**
YAM			76	**7**	8	**14**	**86**
Lily			75	**6**	8	**14**	**86**
ASMOV			76	**7**	7	**14**	**86**

[24] http://www.infotechsoft.com/products/asmov.aspx

In order to show that ontology matching is different from alignment evolution problem, we compare the accuracies of the selected methods in both contexts.

The selected data set was dedicated to compare the accuracies of tools in ontology matching problem. It contains a reference alignment for each test which allows us to measure the precision of each method. In alignment evolution context, we haven't these references alignment. Hence, it's not possible to use the same precision measure. Instead, we use the cardinality based degree of incoherence. The coherence degree measure gives the safe reusing percentage of an alignment after the ontology change. The last column of table 9 gives the accuracy average of each method. Figure 18 summaries this comparison. It shows the harmonic precision of our three tests and the correspondent average coherence degree. Although a high precision of 100% have been registered in the context of ontology matching, it can be considered as the safe reusing percentage in the alignment evolution context.

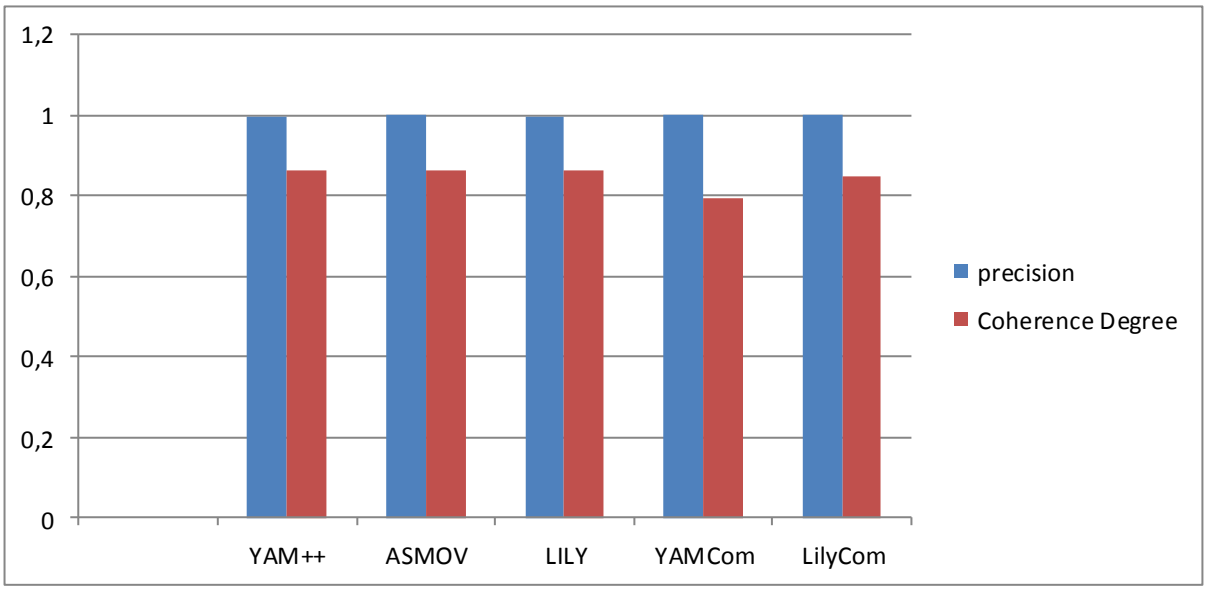

Figure 18: Performances comparison of methods in alignment evolution and ontology matching contexts.

6.4 Conclusion

In this chapter, we discussed the architecture of a prototype implementation of our system of alignment evolution. Our system follows a modular architecture which allows it to scale to the integration of different ontology formats, different alignment interpretations, and others techniques of alignment evolution. Also, we discussed experiences from applying some methods of our framework for demonstrating the limit of some approaches from the category of the adaptive and perfective alignment maintenance approaches. Results show that all selected methods suffer from filling all gaps in the alignment evolution problem. Neither ontology matching nor alignment debugging methods fit well for the alignment evolution problem. While it seems normal for ontology matching tools and debugging methods because there were not designed for this purpose, this is a major drawback for alignment evolution methods. Hence, there is an emergency to invest in dedicated methods. The experimentation also shows the advantage of our approach relatively to these methods.

Chapter 7. Conclusion and future works

7.1 Introduction

This final chapter reflects on the achievements of this dissertation and to look forward to possible new research directions. The reminder of this chapter is structured as follows. Section 7.2 summarizes the work presented in this dissertation. Section 7.3 discusses the main contributions and achievements of this dissertation, thereby reflecting on the problem statement as described in the introduction (see Section 1.2). Finally, Section 7.4 presents possible future works.

7.2 Summary

In this dissertation, we have presented a new approach for the alignment evolution under ontology change problem. This approach proposes a formal framework that consists of a number of phases, each having a specific purpose. The framework facilitates the ontology change identification for maintainers, evolves alignment from a consistent state to another consistent state, conducts to a new consistent state with a minimal of change, and permits to maintainers validating the new alignment by accepting the change, recovery from unnecessary changes, adapting the change, tracking it, or cancelling all the change. In what follows, we review the different phases and the main offered functionalities of this framework.

In this framework, an ontology is a vocabulary and a set of axioms specifying the meaning of this vocabulary. The framework computes the ontology change as the difference between versions of the same ontology. An ontology change is stored as instances of an ontology of change. The main concepts of this ontology are added and deleted vocabulary elements on the one hand and the added and deleted axioms on the other hand. This format of change representation is general enough to encompass any ontology language.

On the light of belief base revision theory, the framework offers different rational operators for evolving alignments. More precisely, the framework adapts the kernel framework of base revision theory to design two general operators: the kernel contraction and the kernel consolidation for alignment evolution. The former designed to deal with the problem of discarding axioms from ontologies. The latter is defined to restore alignment consistency following adding axioms in ontologies implied in alignment. Besides, the framework offers two particular operators based on confidence values associated to alignment correspondences: the confidence based kernel contraction operator and the confidence based kernel consolidation operator. All these operators base their actions on the notions of the alignment kernel and the incision function. The alignment kernel is the set of the minimal subsets of correspondences causing the violation of the alignment consistency. The incision function selects from each element of the kernel at least one correspondence for resolving inconsistencies. For the general operators, no assumptions are made about the incision functions which make the framework very flexible and users are free to choose their own functions. The framework adapts the Hitting set algorithm of diagnosis theory to compute the alignment kernel as well as the corresponding incision functions. As a result, the framework proposes different algorithms with different complexities varying from an exponential to a polynomial time. All the designed operators are characterized by a set of postulates which meet the constraints of the logical consistency, the change preservation, and the minimal change but not the constraint of the structural consistency. The framework is extended with a global method that deals with all types of consistency, namely, the logical consistency, the change preservation consistency, and the structural consistency as well. This method is an orchestration of a set of operations each of which is designed to take care of one aspect of the alignment change process.

Furthermore, the framework manages the alignment change in order to be validated by users. Alignments maintainers may validate the change, recover the unnecessary changes, adapt, track, or cancel the change. For this purpose, the framework stores the change in a journal of change, informs the alignment maintainer about the cost of change, explains inconsistencies, and justifies the proposed change. The notions of kernel and incision function play an important

role to rationalize the interaction between maintainers and the alignment evolution system. An alignment kernel is a set of explanations of alignment inconsistency. Incision functions select among these explanations the accused correspondences to establish consistency. Hence, we can consider incisions functions as change justifications.

Finally, the framework delivers the new alignment and the final associated change in a machine readable format. This allows parsing and exploitation of changes by maintenance tools of depending applications. The final change is the difference between the old and the new delivered alignment.

Finally, the dissertation presents the architecture of a prototype implementation of this framework which constitutes a system for alignment evolution. This system follows a modular architecture which allows it to scale to the integration of different ontology formats, different alignment interpretations, and others techniques of alignment evolution. Also, the dissertation demonstrates the advantage of our approach relatively to some approaches from the category of the adaptive and perfective alignment maintenance approaches. The results show that neither ontology matching nor alignment debugging methods fit well for the alignment evolution problem.

7.3 Contributions

In this section, we discuss the contributions and accomplishments that are the result of this dissertation through the requirements of the alignment evolution problem we formulated in the introduction (see Section 1.2). We begin with our contributions to the methodology knowledge.

Problem 1(ontology change identification): Maintainers want to create their own set of the ontology change in order to understand what happen and correctly update their alignments.

To assist alignment maintainers for creating their own sets of change, the system compares between versions of the same ontology and delivers the change as the changed vocabulary in one hand and the changed axiomatic meaning of this vocabulary in the other hand. This format of change which constitutes an ontology

of change facilitates change sharing not only with the alignment maintainer but also with others tools of the alignment evolution system to automatically manipulate it. Besides, this format of change representation is general enough to encompass any ontology language and makes the change clear and easily understandable.

Problem 2(alignment consistency): As ontologies evolve from a consistent state to another, alignment evolution should follow this change by a transition to a new consistent state. Alignment consistency is expressed as a set of constraints qualified as hard since their violation makes obsolete the alignment and useless.

The framework distinguishes three types of alignment consistency, namely, the logical consistency, the change preservation consistency, and the structural consistency. To resolve the logical consistency and the change preservation consistency the framework adapts the kernel framework of belief base revision theory to design two rational operators: the kernel contraction and the kernel consolidation operators. The kernel contraction changes the alignment to ensure the change preservation consistency in such way that discarded axioms from the aligned ontologies can't be generated again. The kernel consolidation operator restores the logical consistency of an alignment when some added axioms to the aligned ontologies make this alignment inconsistent. To ensure the change preservation consistency, the operator is only authorized to modify the alignment and it can't in any way affect ontologies. These operators base their actions on the notions of the alignment kernel and the incision function. The framework adapts the Hitting set algorithm of diagnosis theory to compute the alignment kernel as well as the corresponding incision functions. Hence, the framework is enough flexible by allowing users to choose among many possibilities of change. For the purpose of satisfying the structural consistency, the framework only suggests correspondences removing.

Problem 3 (minimality of change): In contrast with the consistency constraints which are qualified as hard we qualify the minimal change as a soft constraint. Since the violation of this constraint don't hamper the use of alignments. Hence, we are satisfied by proposing a weak form of this principle. The designed operators for the alignment consistency resolution satisfy the core-retainment postulate which means only correspondences that participate somehow in the

inconsistency implication need to be changed. Sometimes, not all these correspondences should be changed but only a subset of them. This is why we need the user involvement to achieve the operation of alignment change.

Problem 4 (User involvement): alignment evolution is a knowledge intensive task which can't fulfill without the involvement of users. The system proposes a change and maintainers are invited to review it before implementation. Maintainers may validate the change, recover the unnecessary change, adapt, track, or cancel the change. Hence, the system should facilitate the interaction and enhance the interoperability with users.
Besides inconsistency checking, our system of alignment evolution counts on the notions of the kernel and the change log to facilitate the interaction with users and enhance its interoperability with tiers. The former plays the role of inconsistency explanations while the latter allows change tracking and change sharing with applications depending of alignments.

In what concerns the literature review, our contribution is two-fold. First, we have reviewed the main ontology evolution frameworks. Guiding by the fixed requirements of the problem of alignment evolution, we have concluded that these frameworks should be adapted in order to embed the alignment evolution problem. Moreover, we have recommended the separation the study of the alignment evolution problem from the ontology evolution problem since alignment depending artifacts may create confusion with depending artifacts of ontologies. These recommendations have leaded us to propose an alignment change process with fourth phases: a phase for the ontology change identification, a phase for the semantics of change, a phase for the change validation, and a phase for the change implementation. This change process is general and can be concretized in different ways. Besides the different proposed operators of change, the framework is extended with a global method which is an orchestration of a set of operations each of which is designed to take care of one aspect of the alignment change process. Inspired by the classification of the software evolution and maintenance approaches in software engineering, our second contribution is the classification of the alignment evolution approaches in three classes: adaptive, corrective, and perfective maintenance. After review, we observed all approaches fall in two

categories. The approaches of the former are corrective since they check and resolve inconsistences after change. The main challenge for these approaches is how to ensure a consistency alignment with a minimal of change. While the approaches of the latter are adaptive and perfective since they don't consider explicitly the alignment consistency and they only adapt the alignment according to the detected changes in ontologies. Consequently, no guaranties are given to ensure the alignment consistent even they claim it was their primary purpose. This has leaded us to conduct an experimental process in order to demonstrate the advantage of our approach relatively to some approaches from this category. The results show that neither ontology matching nor alignment debugging methods fit well for the alignment evolution problem.

7.4 Perspectives

A major conclusion that can be drawn from these experiences is that the problem of alignment evolution has not received a lot of importance and many investigations must be carried out to solve the issues related to this problem. Investigations should touch the fundamental as well as the methodology aspects of this problem.

Results of this dissertation are within the alignment natural semantics framework. We need further investigations within the alignment contextual semantics (Bouquet et al, 2003). Within the framework of belief base revision theory, we have assumed that ontology languages verify some logical properties such as monotony and compactness. What about non monotone and non-compacted languages?

At the methodology side, our framework can be extended in many ways. We can integrate others operators such as the partial meet contraction and the partial meet contraction consolidation operators. Our framework is limited to alignment revision under ontology change. Always on the light of base revision, we investigate how to deal with the problem of adding and discarding correspondences from alignments. Discarding correspondences from alignments is the subject of alignment debugging approaches. This problem has been identified as an important problem since the early years of semantic web project

development (Meilicke et al., 2009; Qi et al., 2009). We think that the study of this problem is not investigated yet in its right framework. A promise investigation is to apply belief revision theory for the alignment debugging problem.

Furthermore, we hope to extend this study to deal with the problem of restoring the 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. 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. Local inconsistencies may only be solved by local revision of the concerned ontology or alignment while these both operations of revision can be used independently to resolve the global inconsistency. The work of Euzenat (2015) is a first step to understand the revision of the network of ontologies that may help to consider the problem within the framework of base revision theory.

Another related problem is the maintenance of semantic annotations. Annotations express semantic links between documents contents and domain ontologies. Ontology change might decrease the quality of annotations and make them obsolete and useless. Although the recent advances for annotation systems, the maintenance of existing annotations remains under studied (Cardoso et al., 2016).

Bibliographies

Alchourrón, C. E., Gärdenfors, P., & Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *The journal of symbolic logic*, *50*(02), 510-530.

Brachman, R. J., Levesque, H. J., & Reiter, R. (1992). *Knowledge representation and Reasoning*. MIT press.

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*.

Buitelaar, P., Eigner, T., & Declerck, T. (2004). OntoSelect: A dynamic ontology library with support for ontology selection. In *Proceedings of the Demo Session at the International Semantic Web Conference*. Hiroshima,Japan.

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

Cardoso, S. D., Pruski, C., Da Silveira, M., Lin, Y. C., Groß, A., Rahm, E., & Reynaud-Delaître, C. (2016). Leveraging the Impact of Ontology Evolution on Semantic Annotations. In *Knowledge Engineering and Knowledge Management: 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings 20* (pp. 68-82). Springer International Publishing.

Dalal, M. (1988). Investigations into a theory of knowledge base revision: preliminary report. In *Proceedings of the Seventh National Conference on Artificial Intelligence* (Vol. 2, pp. 475-479).

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*.

Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., ... & Sachs, J. (2004). Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management* (pp. 652-659). ACM.

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.

Doyle, J. (1979). A truth maintenance system. *Artificial intelligence*, *12*(3), 231-272.

Dzbor, M., Domingue, J., & Motta, E. (2003). Magpie–towards a semantic web browser. In *International Semantic Web Conference* (pp. 690-705). Springer Berlin Heidelberg.

D'Aquin, M., Gridinoc, L., Angeletou, S., Sabou, M., & Motta, E. (2007). Watson: A gateway for next generation semantic web applications. In Poster session at the International Semantic Web Conference (ISWC 2007). Busan, Korea,

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

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). Heidelberg: Springer.

Fagin, R., Ullman, J. D., & Vardi, M. Y. (1983). On the semantics of updates in databases. In *Proceedings of the 2nd ACM SIGACT-SIGMOD symposium on Principles of database systems* (pp. 352-365). ACM.

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.

Fensel, D. (2001). Ontologies: Dynamic Networks Formally Represented Meaning. Proceeding of the International Semantic Web Working Symposium (SWWS).

Fermé, E., & Hansson, S. O. (2011). AGM 25 years. Journal of Philosophical Logic, 40(2), 295-331.

Flouris, G. (2006). On belief change and ontology evolution. Phd Thesis. Department of Computer Science, University of Crete.

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.

Gärdenfors, P. (1992). *Belief Revision: An Introduction*. In Gärdenfors (ed.), Belief Revision (1–28). Cambridge: Cambridge University Press.

Genesereth, M. R., & Fikes, R. E. (1992). Knowledge interchange format-version 3.0: reference manual. Technical Report Logic-92-1. Computer Science Department. Stanford University, California. http://logic.stanford.edu/kif/Hypertext/kif-manual.html

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. In *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.

Guarino, N., Oberle, D., & Staab, S. (2009). What is an Ontology?. In *Handbook on ontologies* (pp. 1-17). Springer Berlin Heidelberg.

Haase, P., Rudolph, S., Wang, Y., Brockmans, S., Palma, R., Euzenat, J., & d'Aquin, M. (2006). NeOn Deliverable D1. 1.1 Networked Ontology Model.

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

Hansson, S. O. (1994). Kernel contraction. *The Journal of Symbolic Logic*,*59*(03), 845-859.

Hansson, S. (1997). Semi-revision. *Journal of Applied Non-Classical Logics*,*7*(1-2), 151-175.

Hansson, S. O. (1999). *A Textbook of Belief Dynamics. Theory Change and Database Updating*, Dordrecht: Kluwer.

Hansson, S. O., & Wassermann, R. (2002). Local change. *Studia Logica*,*70*(1), 49-76.

Hansson, S. O. (2006). Logic of belief revision. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy*. The Metaphysics Research Lab.Center for the Study of Lan-guage and Information.Stanford University, http://plato.stanford.edu/entries/logic-belief-revision

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.

Hepp, M. (2008). Ontologies: State of the art, business potential, and grand challenges. In *Martin Hepp, Pieter De Leenheer, Aldo de Moor, York Sure. (Eds.): Ontology*

Management: Semantic Web, Semantic Web Services, and Business Applications (pp. 3-22). Springer US.

Horridge, M. (2011). *Justification based explanation in ontologies. Phd Thesis,* School of computer science, University of Manchester.

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

Jean-Mary, Y. R., Shironoshita, E. P., & Kabuka, M. R. (2009). Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, *7*(3), 235-251.

Kalfoglou, Y., & Schorlemmer, M. (2003). Ontology mapping: the state of the art. *The knowledge engineering review*, *18*(01), 1-31.

Kalyanpur, A. A. (2006). *Debugging and repair of OWL ontologies. Phd Thesis,* The Graduate School, University of Maryland.

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.

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

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.

Klein, M., & Fensel, D. (2001). Ontology versioning on the Semantic Web. In *Proceedings of the First International Conference on Semantic Web Working* (pp. 75-91). CEUR-WS. org.

Klein, M., Fensel, D., Kiryakov, A., & Ognyanoff, D. (2002). Ontoview: Comparing and versioning ontologies. In *Collected Posters of First Int. Semantic Web Conf.(ISWC 2002)*.

Kremen, P., Smid, M., & Kouba, Z. (2011). OWLDiff: A practical tool for comparison and merge of OWL ontologies. In *Database and Expert Systems Applications (DEXA), 2011 22nd International Workshop on* (pp. 229-233). IEEE.

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

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..

Lopez, V., Pasin, M., & Motta, E. (2005). Aqualog: An ontology-portable question answering system for the semantic web. In *European Semantic Web Conference* (pp. 546-562). Springer Berlin Heidelberg.

Lopez, V., Motta, E., & Uren, V. (2006). Poweraqua: Fishing the semantic web. In *European Semantic Web Conference* (pp. 393-410). Springer Berlin Heidelberg.

Baader, F., & Nutt, W. (2003). Basic description logics. In *Description logic handbook* (pp. 43-95). Cambridge University Press.

Baader, F., Penaloza, R., & Suntisrivaraporn, B. (2007). Pinpointing in the Description Logic EL+. In *Annual Conference on Artificial Intelligence* (pp. 52-67). Springer Berlin Heidelberg.

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

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

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. (2008). Incoherence as a basis for measuring the quality of ontology mappings. In *Proceedings of the 3rd International Conference on Ontology Matching-Volume 431* (pp. 1-12). CEUR-WS. org.

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.

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

Nebel,B. (1994). Base Revision Operations and Schemes: Semantics, Representation, and Complexity. In *Proceedings of the 11th European Conference on Artificial Intelligence (341-345*). Amsterdam, the Netherlands: John Wiley & Sons

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

Noy, N. F. (2009). Ontology mapping. In *Handbook on ontologies* (pp. 573-590). Springer Berlin Heidelberg.

Noy, N. F., Fergerson, R. W., & Musen, M. A. (2000). The knowledge model of Protege-2000: Combining interoperability and flexibility. In *International Conference on Knowledge Engineering and Knowledge Management* (pp. 17-32). Springer Berlin Heidelberg.

Noy, N. F., Chugh, A., Liu, W., & Musen, M. A. (2006). A framework for ontology evolution in collaborative environments. In International semantic web conference (pp. 544-558). Springer Berlin Heidelberg.

Noy, N. F., & Klein, M. (2004). Ontology evolution: Not the same as schema evolution. *Knowledge and information systems*, *6*(4), 428-440.

Noy, N. F., & Musen, M. A. (2002). Promptdiff: A fixed-point algorithm for comparing ontology versions. *AAAI/IAAI*, *2002*, 744-750. Edmonton, Alberta, Canada

Noy, N. F., & Musen, M. A. (2003). The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, *59*(6), 983-1024.

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.

Oliver, D. E. (2000). Change Management and Synchronization of Local and Shared Versions of Controlled Vocabulary. *Phd Thesis.*

Palma, R., Haase, P., Corcho, O., & Gómez-Pérez, A. (2009). Change representation for OWL 2 ontologies. In *Proceedings of the 6th International Conference on OWL: Experiences and Directions-Volume 529*(pp. 142-151). CEUR-WS. org.

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

Papavassiliou, V., Flouris, G., Fundulaki, I., Kotzinos, D., & Christophides, V. (2009). On detecting high-level changes in RDF/S KBs. In*International Semantic Web Conference* (pp. 473-488). 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.

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

Qi, G., Ji, Q., & Haase, P. (2009). A conflict-based operator for mapping revision. In *International Semantic Web Conference* (pp. 521-536). Springer Berlin Heidelberg.

Redmond, T., & Noy, N. (2011). Computing the changes between ontologies. In *Joint Workshop on Knowledge Evolution and Ontology Dynamics* (pp. 1-14).

Ribeiro, M. M., & Wassermann, R. (2007, June). Base revision in description logics-preliminary results. In *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)* (Vol. 6982).

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial intelligence*, *32*(1), 57-95.

Rymon, R. (1991). A Final Determination of the Complexity of Current Formulations of Model-Based Diagnosis (Or Maybe Not Final?). Technical Report No. MS-CIS-91-13.University of Pennsylvania.

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.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, *5*(2), 51-53.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: principles and methods. *Data & knowledge engineering*, *25*(1-2), 161-197.

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

Swanson, E. B. (1976). The dimensions of maintenance. In*Proceedings of the 2nd international conference on Software engineering* (pp. 492-497). IEEE Computer Society Press.

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

Wang, P., & Xu, B. (2008). Debugging ontology mappings: a static approach.*Computing and Informatics*, *27*(1), 21-36.

Zahaf, A. (2012). Alignment Between Versions of the Same Ontology. In ICWIT (pp. 318-323).

Zahaf, A., & MALKI, M. (2016a). Kernel Contraction and Consolidation of Alignment under Ontology Change. Journal of Information Technology and Computer Science(IJITCS), 8(8), 31-42.

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