Artif Intell Rev (2017) 47:181–215
DOI 10.1007/s10462-016-9479-5

Merging of axiomatic definitions of concepts in the complex OWL ontologies

Muhammad Fahad[1]

Published online: 25 April 2016

Abstract In the last decade, ontology matching and mapping research has shown a measurable progress. This topic draws substantial attention within the research community, though it is not fully researched so far and new complex and effective solutions are needed. Current works are limited in finding alignments or mappings between concepts of heterogeneous ontologies. But, once ontology mappings are found, then how they (or their class expressions) are to be integrated *automatically* is left open for the ontology merging research. This paper elaborates the mapping of class expressions of concepts and contributes an algorithm for their merging in an automatic ontology merging process *without any human intervention*. However, the challenge of mapping axiomatic definitions is the most difficult task for merging concept definitions of the source ontologies, but it reveals significant increase in precision and recall values. In addition, with the study of these algorithms, we conclude that ontology merging facilitates when one wants to get ontology with the better quality as the *combined rich axioms* are added in the merged ontology. We also discuss the results of our first successful participation in the Conference, OA4QA and Anatomy track of OAEI 2015.

Keywords OWL ontology · Axiomatic definitions · Ontology mapping and merging · Heterogeneity · Interoperability

1 Introduction

The semantic information gathering from heterogeneous information sources on the World Wide Web is a hot research challenge for the entire web community (Arch-Int and Sophatsathit 2003). *Ontology Merging*, as a specialized task of information integration, got a huge attention in the last decade. The identification of mappings is the fundamental step to tackle

✉ Muhammad Fahad
mhd.fahad@gmail.com
http://www.disp-lab.fr/

[1] Decision and Information Sciences for Production Systems (DISP) Lab, 160 Boulevard de l'université, 69676 Bron, France

the heterogeneity and plays a vital role in the ontology merging process. On the basis of identified mappings, candidate merge operations between mapped concepts are performed to achieve the merged ontology. In the ontology-based research literature, there is a wide range of ontology matchers and mappers with diverse techniques. According to Alasoud et al. (2009), an end-user is not sure about the suitability of a mapping technique without any prior knowledge in his application and the combinations of different techniques may benefit more than the individual ones. A detailed and recent survey on the state-of-the-art ontology matching is provided by the Shvaiko and Euzenat (2012). Although, ontology matching and mapping research has shown a measurable progress, these works are limited in finding only alignments or mappings between ontologies. Once the mappings are found, then *how they (or their class expressions) are to be integrated automatically is left open for the ontology merging research*. In addition, ontology merging is a multi-steps (at least 2 steps) process, and the matching/mapping is only one part of it (Euzenat and Shvaiko 2007).

Achieving accuracy of the merged output is the major challenge of ontology merging research. Regardless of few ontology mapping systems (Jiménez-Ruiz and Cuenca-Grau 2011; Huber et al. 2011), current ontology merging systems do not follow appropriate mechanisms for achieving the accuracy, consistency and conciseness of the merged ontology (Fahad et al. 2011). Semi-automatic systems (such as *Prompt* Noy and Musen 2003, *Chimaera* McGuinness et al. 2000, *MoA* Dou et al. 2002, etc.) depend on end-users for the validation of mappings and need human intervention in the decision making process of merging concepts. These systems do not propose a consistent list of suggestions and do not highlight the contradictory elements between the ontologies, which is a tedious and time consuming task for the end-user. Automatic systems (such as *OM* Guzman-Arenas and Cuevas-Rasgado 2010, *Atom* Raunich and Rahm 2011, etc.) are capable of producing the merged ontology on-the-fly, but do not tackle the complexity of axiomatic OWL definitions automatically. These systems benefit from the merging of light-weight ontologies and end-users can only merge their class hierarchies. For the former limitation, in our previous research, we contributed and embedded a *Quality Mechanism* (based on *Consistency, Completeness and Conciseness*) inside an automatic merging system DKP-AOM that captures all the information from the input ontologies and provides a full merge solution (Fahad et al. 2011, 2012). The embedded quality criteria remove the load of validation on an end-user and ontology merging becomes facilitated with automatic validation and verification of the resultant output. The proposed system highlights the contradictory elements between the ontologies, however, this aspect is out of the scope of this paper and can be found in Fahad et al. (2011, 2012). For the latter limitation, in this paper we elaborate our mapping criteria for the detection of all possible mapping pairs by analysing their class expressions (i.e., axiomatic definitions) of concepts and then merge their *individual axiomatic expressions to form a combined expression automatically* to improve the effectiveness of our merging system DKP-AOM.

Effectiveness (i.e., to detect all possible mapping candidates) of an automatic ontology merging system is an important issue to address. It removes the load of manual checking of candidate concepts for merging by an end-user and also eliminates the possibility of redundancy in an automatic merged ontology. Otherwise, undetected mapping candidates compromise the conciseness of the merged output. Therefore, mapping of class expression of concepts is crucial task to detect those concepts which are defined by the different terminological names but are represented by semantically same class expressions. Although the automatic decomposition and analysis of axiomatic definition is a very complex task, but is very challenging for the advancement of automatic ontology merging research.

For finding axiomatic mappings, first it is necessary to compute basic mappings of concepts and properties. For this purpose, we benefit from the ontology mapping researches

(Shvaiko and Euzenat 2012). But, most of the current works in ontology merging literature employ string, synonym and linguistic techniques to tackle various types of semantic heterogeneities in the semantic web ontologies. We also adapted natural language processing (NLP) Techniques by using *MorphAdnorner* (Morphadorner 2010) before applying these basic techniques. But when concepts are represented by the different terminology and defined with the semantically equivalent class expressions, then the current works fail to merge such axiomatic mappings between concepts. Therefore, one of the requirements for today's ontology merging research is to focus on the *axiomatic mapping and merging of concepts* to bridge this gap. OWL axioms have clear semantics and syntax and play vital role in describing the semantics of the domain knowledge in ontologies. For example, a subclass axiom is described with an atomic concept on the left hand side and a class expression (typically a property restriction) on the right hand side (Bechhofer et al. 2004). For example, in OWL2 (Bock et al. 0000) ontologies, the *ObjectSomeValuesFrom* class expression allows for the existential quantification over an object property, and it contains those individuals that are connected through a property expression to at least one instance of a given class expression. However, the challenge of mapping and merging axiomatic definitions is the most difficult task for merging concept definitions of the source ontologies. But, this is obvious that concepts with different names are not identified by the string, linguistic or synonym strategies without considering their class expressions. Therefore, the role of axiomatic analysis of concepts with class expressions becomes active. Otherwise, such concepts are not identified by the matching module of the merging system, and they compromise the precision of mappings and create redundancies in the merged ontology. This paper contributes this idea of axiomatic definitions in detail so that all possible mapping candidates are detected and effectiveness of the merging system is enhanced. This idea is really helpful for the users who are not familiar with OWL axioms, but want to construct their ontology automatically by reusing existing ontologies.

The rest of the paper is organized as follows. Section 2 discusses the background. Section 3 throws a light on related works of this domain. Section 4 develops our main symbolic calculation while computing mappings between the axiomatic definitions of concepts. Section 5 contributes algorithms for the merging of axiomatic definitions (or class expressions) in which concepts are expressed in the context with other concepts and properties. In Sect. 6, we presented our developed test cases for the mapping and merging modules of our system. In Sect. 7, we build a comparative analysis of our system with the state-of-the-art. Section 8 present the results of our first successful participation in the Conference, OA4QA and Anatomy track of OAEI 2015. Section 9 concludes the paper and shows future directions.

2 Background

2.1 Terminology versus axiomatic ontologies

This section discusses about terminological ontologies and axiomatic ontologies, which is important to know in the scope of this paper. In the ontology-based research literature, Sowa (1996) classifies ontologies into two main groups based on the amount and type of structure of conceptualization that they capture. These two groups are Terminological Ontologies and Axiomatic Ontologies. First, *Terminological Ontologies* are ontologies that capture concepts which are partially specified in a hierarchy by parent-child relationships, but do not capture sufficient knowledge in terms of axioms and definitions. Such ontologies do not have necessary and sufficient conditions for the concept definitions and thus lack powerful reasoning and inference mechanisms. These ontologies are also referred as *Lexical or Light-Weight* ontologies. These are simple to create, evaluate and merge. Second, *Axiomatic Ontologies* are terminological ontologies that capture concepts which are fully defined with necessary and sufficient conditions that can be automatically translated to logic. Such ontologies are logic oriented and support complex inferences and computations, and often called as *'complex ontologies'*. These ontologies are also referred as *Formal or Heavy-Weight* ontologies. They are difficult to engineer and require complex algorithms for their evaluation and merging. In the context of this paper, we are working with these ontologies and take an initiative to address the most difficult task of mapping and merging of their concepts.

2.2 Example of axiomatic mapping and merging of concepts

Mapping and merging of *axiomatic definitions* to produce merged output automatically is really a very complex task. Therefore, this section elaborates the idea behind their mapping and merging. Consider two conference ontologies O1 and O2. The concept *Article* in an ontology O1 and the concept*Paper* in an ontology O2 are not identified as the candidates for merging by basic strategies. But, if we analyse their axiomatic definitions *[O1:Article subclassOf {document and (writtenBy some author) and (has_author min 1)}]* and *[O2:Paper subclassOf {doc. and (hasAuthor min 1) and (is_written_by some author)}],* then these concepts are detected to be the candidates of merging as they have same axioms built with semantically same concepts and properties. Similarly, consider a concept 'AbstractPaper' with two different definitions. In ontology O1, it is a defined by a restriction of type *AllValuesFrom, markedBy only GuestReviewer.* In ontology O2, it is defined by a restriction of type *AllValuesFrom, reviewedBy only PCMember*. The merging system should be capable enough to merge their definitions to produce a combined merged definition as depicted in the Table 1. In this paper, we propose algorithms to form such combined merged axiom from the individual definitions from the source ontologies. Later in this paper, we elaborate these algorithms for the generation of such axiomatic definitions.

3 Related work

Ontology mapping is the primary step for the ontology merging, therefore, we discuss the most recent description logic based ontology mappers which participated in the OAEI 2015 with our system DKP-AOM. AgreementMakerLight (AML) is an element-level matcher embedded with the lexical matching techniques and also based on the external resources (i.e., background knowledge) (Cruz et al. 2009). In addition, it also computes similarity between two classes by propagating the similarity of their matched ancestors and descendants. It is developed for biomedical ontologies but now considered as a general-purpose matching system as well. It has been participating in OAEI for many years and achieved best performance. LogMap is another scalable and logic-based ontology matching system (Jimenez-Ruiz and Cuenca Grau 2011). It has various variant, i.e., LogMapBio, LogMapC, LogMapLt. It is a stable and mature system that uses ontology modules with well-understood semantic properties to efficiently compute mappings. XMap is another ontology matching system whose semantic similarity measure has been defined using UMLS and WordNet to provide a synonymy degree between two entities from different ontologies. Its algorithm is based on both the lexical and structural context on ontological elements (Djeddi and Khadir 2014). GMap ontology mapping system is an alternative probabilistic scheme which com-

Table 1 Example of axiomatic merging of concepts

Ontology 1: ‘AbstractPaper’ concept	Ontology 2: ‘Abst_Paper’ concept
<owl:Class rdf:ID=“AbstractPaper”>	<owl:Class rdf:ID=“Abst_Paper”>
<rdfs:subClassOf>	<rdfs:subClassOf
<owl:Restriction>	rdf:resource=“http://www.w3.org/2002/07/owl#Thing”/>
<owl:onProperty>	<rdfs:subClassOf>
<owl:ObjectProperty rdf:ID=“markedBy”/>	<owl:Restriction>
</owl:onProperty>	<owl:onProperty>
<owl:allValuesFrom rdf:resource=“#GuestReviewer”/>	<owl:ObjectProperty rdf:ID=“ReviewedBy”/>
</owl:Restriction>	</owl:onProperty>
</rdfs:subClassOf>	<owl:allValuesFrom>
<rdfs:subClassOf rdf:resource=“http://www.w3.org/2002/07/owl#Thing”/>	<owl:Class rdf:about=“#PCMember”/>
</owl:Class>	</owl:allValuesFrom>
	</owl:Restriction>
	</rdfs:subClassOf>
	</owl:Class>
Ontology 3: ‘AbstractPaper’ merged concept with axiomatic definition	
<owl:Class rdf:ID=“AbstractPaper”>	
<rdfs:subClassOf rdf:resource=“http://www.w3.org/2002/07/owl#Thing”/>	
<rdfs:subClassOf>	
<owl:Restriction>	//from ontology 1
<owl:onProperty rdf:resource=“ #markedBy”/>	
<owl:allValuesFrom rdf:resource=“#GuestReviewer”/>	
</owl:Restriction>	
</rdfs:subClassOf>	
<rdfs:subClassOf>	
<owl:Restriction>	//from ontology 2
<owl:onProperty rdf:resource=“ #ReviewedBY”/>	
<owl:allValuesFrom rdf:resource=“ #PCMember”/>	
</owl:Restriction>	
</rdfs:subClassOf>	
</owl:Class>	

bines the sum-product network and the noisy-or model (Li and Sun 2015). It uses sum-product network to encode the similarities based on individuals and disjointness axioms. The noisy-or model is utilized to encode the probabilistic matching rules, which describe the influences among entity pairs across ontologies. Although, GMap got reasonable precision and recall but it did not consider mapping repair techniques. Only AML, LogMap, XMap and DKP-AOM use repair techniques and achieve coherent results for the ontology mappings, which is also evident by the results of Ontology Alignment for Query Answering track of OAEI.

In the research literature, there are two broad categories of ontology merging approaches (Bruijn et al. 2006). In the first approach, merging process results a *single output ontology* that contains the individual source ontologies. The examples of this approach are *Prompt* (Noy and Musen 2003), *Chimaera* (McGuinness et al. 2000), etc. These systems use terminology on the basis of string measures to propose suggestions for the end-users, which are later on used for merging ontologies by following a cyclic semi-automatic process with the high level of end-user dependency. In the second approach, merging process results a *bridge ontology* that imports the source ontologies and includes bridge axioms or articulation rules that represent the mappings about the concepts of the source ontologies. The examples of this approach are OntoMerge (Dou et al. 2002), ONION (McGuinness et al. 2000), etc. Another multi-technique system named as MoA (Dou et al. 2002) follows a hybrid mechanism adopting features of both approaches. Its intermediate output is related to ONION (Mitra and Wiederhold 2002) as it produces articulation rules, and final output as a new merged ontology like other semi-automatic system Prompt (Kim et al. 2005).

Ontology merging can be classified as a symmetric or asymmetric approach (Raunich and Rahm 2012). Early works for ontology merging (such as *Prompt, Chimaera, OntoMerge,* etc.) exploit a symmetric solution. They treat all the input ontologies with the same priority level during the merging process and the output ontology may or may not resembles with any of input ontologies. However, recent solutions such as *OM* (Guzman-Arenas and Cuevas-Rasgado 2010) and *Atom* (Raunich and Rahm 2011) exploit an asymmetric approach. In the asymmetric solution, the merging process prioritizes one of the input ontologies and merges the other input ontology within the prioritized ontology keeping safe its structure. As an example, in the asymmetric solution when an ontology O1 has a priority, then the structure of an ontology O1 is preserved in the merged ontology O3. For instance, the concept D in O1 is maintained in O3 as a sub-concept of B, regardless of its structural conflict with O2 where it is sub-concept of E.

Besides these merging techniques and system, there is an instance-based semi-automatic merging methodology exploited by FCA-MERGE (Stumme and Mädche 2001). It takes the source ontologies and text documents that comprise of instances, and adopts a bottom-up technique for ontology merging. It matches the application specific instances and when finds common instances, then concepts that instantiate them are taken as candidates for merging. A major drawback of instance based methodology is observed when semantically different concepts are merged on the basis of having same instances.

Another category of merging tools is based on either existing mapping tools or DL Reasoners. Jimenez-Ruiz et al. (2009, 2008) contributed a safe and logic based methodology for the modular reuse of ontologies. Their tool provides a general and flexible method to facilitate the integration of heterogeneous ontologies by using existing mapping tools (such as OLA, CIDER, AROMA). They assume that the reuse is performed by simply building the logical union of the axioms in the modules under the standard semantics. Maiz et al. developed an approach for the automatic ontology merging based on the hierarchical clustering and inference mechanisms (Maiz et al. 2010) by using description logic services of DL-Reasoner *Pellet* to find all possible relations between the concepts of ontologies. Besides these, HCONE-Merge uses linguistic and structural knowledge about ontologies to formalize the Latent Semantics Indexing (LSI) method (Kotis et al. 2009). Then, it makes use of the LSI mechanisms for computing all the possible correspondences by mapping intended informal meaning of concepts onto *Wordnet* senses, and then requires an end-userfor their validation.

There is a tool named *IMerge* that focuses on the presentation during the merging of ontologies.*IMerge* provides different visualization facilities that aid in the process of semi-automatic merging of ontologies (Jerroudi and Ziegler 2008). The SmartTree-View displays the hierarchical structure and provides options for the exploration and development of ontologies. By Matrix-View, one can compare two ontologies and visualization displays the possible mappings. Finally, the Merge-View provides an option to merge ontology with the user feedback by accept, change or reject options for each of the candidate Merge options. The merge candidate pair concepts are detected based on the string, structure and addition input document that has annotations to the concepts of ontologies. Other than these techniques, there is a tree-structure based ontology integration, which is designed to map concepts of an ontology to a tree structure with the help of attribute matching between concepts (Xie et al. 2011). Once they are mapped, this technique restructures the tree, and relies on the tree structure to integrate the source ontologies. The purpose of such integration is to answer different queries, rather than to generate a new merged ontology.

The above mentioned merging techniques can be applied according to the requirements of the merging task. Therefore, it has been agreed upon by the researchers that for a merge process, there is no single best solution (Raunich and Rahm 2012). It depends on the requirements of the task where and when merging process is executed. However, in any merging approach, the main hurdle lies in the identification and the resolution of semantic heterogeneities. Semantic heterogeneities occur due to the differences in the domain interpretation and modelling of the knowledge within the ontologies, giving rise to various types of mismatches and conflicts (Pottinger and Bernstein 2003). Visser et al. (1997) provide an analysis of ontological mismatches and heterogeneities that may belong to language level and ontology level. Language heterogeneity or language level mismatches happen when ontologies that were developed in different languages are merged, as they differ in syntax, expressiveness, semantics of primitives, etc.

The current major challenge faced by the ontology merging research is to develop *an automatic consistent ontology merging solution* that identifies all the mismatches, produces reliable and usable merging results, and handles the merging of complex ontologies such as OWL 2 (W3C current recommendation). Current systems handle the merging of light-weight ontologies and do not tackle the complex constructs of OWL2 ontologies that comprise concepts which represent real world objects. Therefore, it is a requirement of the current ontology merging research to propose solutions for building merged class expressions of concepts for the merging of OWL2 ontologies to meet the demands of the emerging semantic web. In addition, it should be a *fully automatic solution* so that it is carried-out by the machines, because it cannot be done manually beyond a certain degree of size, number and complexity of ontologies due to the inherent complexity of axioms. The contribution presented in this paper fills this gap by presenting our automatic ontology merging (AOM) system DKP-AOM for fully automatic merging and its novel algorithms implemented in OWLAPI 3 (Horridge and Bechhofer 2011) with the aim of tackling OWL 1.0 and OWL 2.0 rich ontologies that produce output in OWL and other formats (N3, Turtle) as well. Mainly it provides in depth study about the mapping and merging of axiomatic definitionsof concepts by analyzing their class expressions in our merging system.

4 Mapping of axiomatic definitions of concepts for ontology merging

This section develops how we tackle and map axiomatic definitions of concepts in the complex OWL ontologies.

4.1 Basic matchers before axiomatic analysis

4.1.1 Concept label similarity

For mapping axioms, it is a fundamental task to find mappings between all primitive concepts and properties. Therefore, we employ several basic matchers to find mappings between concept and properties. Concept label similarity ($\mathrm{Sim_{lab}}$) computes linguistic and synonym based correspondences between the labels of concepts c and c' of ontologies O_a and O_b. Linguistic (or lexical) similarity finds the string-based correspondences based on *SimMetric* (SimMetrics 2011) between labels. Synonym similarity is computed based on the lexical database *Wordnet* (Miller 1995) that helps to detect the concepts which have the same meanings but are lexically different. For example, concepts that are synonyms (e.g., c_1:*Student*, c_2:*Scholar*) and abbreviations (e.g., c_1:*InformationTechnology*, c_2:*IT*) are determined by this way.

4.1.2 Inheritance based similarity

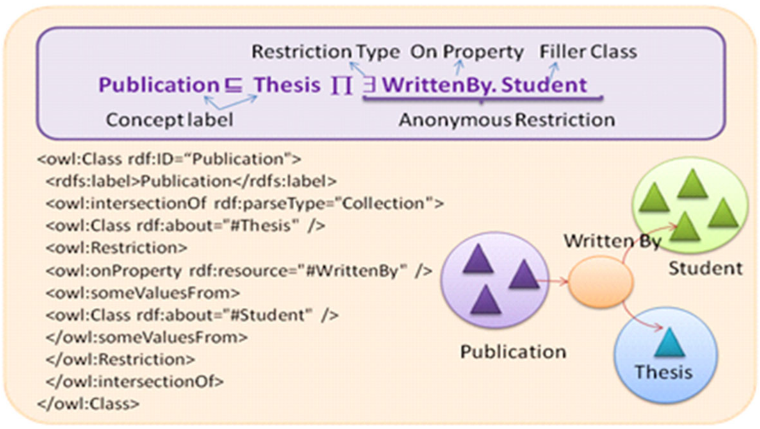
We also considered that an inheritance is a vital factor to detect the mappings candidates for the merging of concepts between source ontologies. It increases a level of confidence that the detected mappings have not just a lexical similarity, but a real mapping having parent-child relation. *Inheritance* matching is done after the *Concept label* similarity and*Synonym* similarity. Consider a scenario, where an ontology O1 contains Person and PhD concepts. Person is defined as: {*Person subclassof hasName some String*}, and another concept PhD is defined as a subconcept of Person concept as: {*PhD subClass Person*}. In ontology O2, there is a PhD candidate that is defined as: {*PhD subclassof hasName some String*}. In such a case we get the basic mappings between O1:PhD, and O2:PhD. Then the inheritance matcher plays an important role by matching the inheritance of the restriction from Person to PhD from O1 with the restriction in O2 and adds the confidence level of their similarity. In this way, an inheritance matching has a potential impact in the proposed solution. This similarity will help for the detection of axiomatic similarities between concepts.

4.1.3 Datatype and object property similarity

Similarly, correspondences between Datatype and Object properties are identified on the basis of their label similarities. Datatype and Object properties in an ontology represent the context and the semantics of concepts. Generally, datatype properties are called the attributes of a concept in the ontology. For example, each *Book* has some attributes such as *ISBN*, *Name*, *Price*, etc. Object properties or relations make direct and reciprocal links between concepts within an ontology. Object properties *Contributes*(*Author*, *Paper*) and *WrittenBy*(*Author, Article)* make associations between the concepts and represent the real descriptions, which help the merging algorithm to judge the real mapping in another ontology. Once these mappings are found between concepts and properties, the desirable task of axiomatic definition is achieved, which is elaborated in the next section.

4.2 Concept DL axiomatic similarity ($\boldsymbol{Sim_{axm}}$)

OWL classes are described through the class descriptions/expressions that enrich the background information of the concepts and represent the constraints of real world situations. For finding the accurate semantic similarity between the concepts of ontologies, DL axioms



```
<owl:Class rdf:ID="Publication">
 <rdfs:label>Publication</rdfs:label>
 <owl:intersectionOf rdf:parseType="Collection">
 <owl:Class rdf:about="#Thesis" />
 <owl:Restriction>
 <owl:onProperty rdf:resource="#WrittenBy" />
 <owl:someValuesFrom>
 <owl:Class rdf:about="#Student" />
 </owl:someValuesFrom>
 </owl:Restriction>
 </owl:intersectionOf>
</owl:Class>
```

Fig. 1 Complex concept axiomatic definition made by intersection between primitive concept and anonymous restriction class. This function also demonstrates the code along the steps of our algorithm for the help of ontology developers

can help significantly as they define the context of the concepts. They link the concept by different means that depict the concept's real semantics. An OWL ontology also supports unnamed classes that are formed by the set of restrictions on the values for particular properties of the concepts. Such class descriptions are equivalent to the description logic (DL) axioms, e.g., a *Publication* concept can be represented as $\{Thesis \prod WrittenBy.Student\}$ or $\{Paper \prod ReviewedBy.Committee \prod haslimit.Pages{>}8\}$ accordingly to its context. Figure 1 shows the owl ontology syntax with an equivalent DL axiom of *Publication* concept.

4.2.1 Types of DL axioms

Axiomatic definitions can be formed from the union, complement, intersection and restriction operators applied on the primitive concept or the anonymous concept and/or by their boolean combinations. Some of the boolean combinations of primitive and anonymous restrictions with different operators. The DL axiom similarity checks the expression of concept descriptions formed by other concepts (primitive/anonymous/combination) and operators between them. We divide these concept descriptions into four cases as follows:

Restriction based concept description Restriction analysis for the semantic similarity between concepts is highly significant as it defines the necessary conditions or necessary and sufficient conditions for them. The necessary condition of a class makes that class a subclass of the restriction class. In case of necessary and sufficient condition for the class, both the restriction class and the restricted class will be interpreted as equivalent, i.e., they always have exactly the same members. For matching anonymous concepts, the similarity constitutes of correspondences between the restriction type (e.g., *SomeValuesFrom, AllValuesFrom, HasValue*), an object property involved and range concepts that act as filler classes in the definition.

Example 1: Let Teacher concept in two ontologies be defined as follows.

O1: $\text{Teacher} \supseteq \exists\ \text{Teaches . Subjects}$
O2: $\text{Teacher} \supseteq \exists\ \text{Teaches . Courses}$

Example 2: Let Accepted_paper be defined in various source ontologies as below.

O1: Accepted_paper $\supseteq \exists$ AcceptedBy . Reviewer
O2: Accepted_paper $\supseteq \exists$ is_Accepted_by . Reviewer
O3: Accepted_paper $\supseteq \exists$ AcceptedBy . (Reviewer U ProgramCommittee)
O4: Accepted_paper $\supseteq 3 \leq$ AcceptedBy_Reviewer
O5: Accepted_paper $\supseteq$ hasDecision = PositiveReview

Conflicts in the restriction-based classes can occur between (1) restriction types as indicated in the example 2 during the matching of Accepted_paper in O1 and O4, (2) Object property involved in the construction of OWL class as indicated in example 2 during matching of Accepted_paper in O1 and O2, and (3) Filler classes involved in the construction of OWL class as indicated in example 1 and example 2 during the matching of Accepted_paper in O1 and O3.

Operator based concept description Context of concepts can be made by different operators applied on the primitive concepts. Such a class description can be in the form of union, intersection or complement between concepts. Matching such concept descriptions need to match operator and operand concepts.

Example 1: Let concept 'Chairman' of conference consists of many people be defined as:

O1: Chairman = ProgramCommittee U ConferenceChair
O2: Chairman = AssociatedChair U ConferenceChair U ProgComChair

Example 2: Let concept 'Presenter' of conference paper be defined as:

O1: Presenter = Author $\prod$ RegisteredParticipant
O2: Presenter = Author $\prod$ RegisteredParticipant $\prod$ Guest_Speaker

Conflicts in the operator-based class description can occur between the operands used in the explication of the context of a concept. For example, operands used between union operators, as indicated in example 1 during the matching of the class description of 'Chairman' concept. In addition, operands used between the intersection operators, as indicated in example 2 during the matching of the class description of 'Presenter' concept.

Boolean combination based concept description Complex ontologies may contain boolean combinations of the above two constructs during the formalization of concept descriptions of the real world objects. It means definitions comprise arbitrary Boolean combinations of classes and restrictions to represent constraints according to real world objects. For example, one can state that a class *'Temporary_Staff'* contains objects that are either *PhDStudent* or *Visiting_Lecturer* and does not comprise *Permanent_Faculty*. Matching such concept descriptions need to match all the boolean combinations, and require much attention for their combined merged definition.

Example 1: Let concept 'Professor' be defined in ontologies as:

O1: Professor = $\exists$ teaches course $\prod \exists$ courseLevel $\neg$ BS_level
O2: Professor = $\exists$ teaches MS_Course $\prod$ level=>5 CourseLevel

Example 2: Let concept 'External Reviewer' be defined in ontologies as:

O1: External Reviewer = Person $\prod$ reviewPaper $\leq$ 2 FullPaper $\prod \exists$ choosenBy PC_Chair
O2: External Reviewer = Person $\prod \neg$ CommitteeMember $\prod \exists$ Rate_paper $\neg$ Accept_paper $\prod$ $\exists$ writeReview Review
O3: External Reviewer = Organizer $\prod$ executes $\exists$ DoubleBlindReview

Conflicts in the class description that are based on boolean combinations can occur between the operands and operators used in the modelling of the context of a concept. These conflicts are in combinations of above conflicts that occur due to mismatches in the restriction or operator-based class descriptions.

Pattern: value partition or enumeration Value Partition pattern formulates a set of values which can hold for a particular property and that are used in a restriction to describe a particular aspect of an OWL concept. The enumeration can be used for the same purpose, but, it contains a set which cannot be further expanded, i.e., no sub-partitioning is possible.

Example 1: Enumeration or Value Partition of 'Review-Values' can be defined in ontologies as:

O1: Review-Values {Accept, Borderline, Reject}
O2: Review-Values {Accept{clear, marginal, weak}, Reject{strong, marginal, weak}}
O3: Review-Values {ReviewerKnowledge {high, low, moderate}, Recommendation {clear_accept, marginal_accept, marginal_reject, reject}}

Mismatches can occur among these definitions. For example, Enumeration of O1:Review-Values can resemble with the Value Partition of O2:Review-Values or O3:Review-Values.

4.2.2 *Calculating axiomatic similarity*

When class descriptions are of the same type (i.e., both are restriction-based, or same operator-based or enumerations), DL axiom similarity tokenizes the class description into the set and performs matching between the operands, i.e., the elements of the set. Thus, the semantic similarity between DL axioms is calculated from the number of matches between the element of two sets (S1 and S2) that belong to the concepts c and c' respectively. The devised formula for the axiomatic similarity calculation is based on Jaccard Similarity string matching algorithm as follows.

$$\text{Sim}_{axm} = \frac{|(\text{S1} \cap \text{S2})|}{|\text{S1 U S2}|}$$

Most heterogeneous ontologies can also be defined by different elements (concepts or operands), operators, and restrictions. In case of a complex DL axiom that comprises boolean combinations, it is required to tokenize the descriptive expression into the basic expressions with primitive concepts. Then, the basic expressions are matched together and finally their aggregated similarity is computed. For instance in the above mentioned example Professor concept, the whole axiom is divided into intersection of two basic axioms, i.e., $\exists$ teaches course and $\exists$ courseLevel $\neg$ BS_level. Then each basic axiom requires matching of restriction type (e.g., someValuesFrom), object property involved (e.g., teaches) and Filler classes (e.g., course). Once the similarity of basic axioms is computed then their values are aggregated to calculate their combined similarity for the whole DL axiom.

The following section shows an example that illustrates how individual matchers (i.e., label and axiom matcher) produce similarity values and how they are aggregated to represent the combined similarity value for a concept from the source ontologies.

4.2.3 *Similarity aggregation*

Based on the previous sections, we implemented many similarity computation matchers (e.g., label matcher, property matcher, axiom matcher, etc.) for finding the correspondences between the concepts of ontologies. When similarities between the concepts of source ontologies are computed, aggregation is performed to find the combined representative similarity value for a concept based on all the types of similarity values produced by the individual matchers. There are many methods to get an aggregation of individual similarities.

One of the simplest methods (i.e., combination) is to get the values from the individual matchers and select the maximum value. This maximum similarity value is the representative of an aggregative similarity value of a concept. For instance, consider an example of enumerated class *ContributionType*$_{O1,O2}$ in the source ontologies below.

Example: Ontology O1:ContributionType *oneOF*{AbstractPaper, PositionPaper, ConferenceFullPaper}
Ontology O2:ContributionType *oneOF*{AbstractPaper, ConferenceFullPaper }

For these concepts with expressions, a string-based Label matcher ($\mathrm{Sim}_{\mathrm{lab}}$) produces a similarity value equal to 1, as their labels (i.e., O1:ContributionType, O2:ContributionType) completely match in the ontologies. An Axiom matcher (Sim_{axm}) based on Jaccard's Measure produces 0.667 as the similarity value. The calculations are:

$$\mathrm{Sim}_{\mathrm{lab}} = 1, \mathrm{Sim}_{\mathrm{axm}} = |\mathrm{S1} \sqcap \mathrm{S2}|/|\mathrm{S1} \cup \mathrm{S2}| = 2/3 = 0.667.$$

For the aggregation of similarity values, the maximum value (i.e., 1) is chosen as the best representative similarity value between $\mathrm{ContributionType}_{\mathrm{O1,O2}}$ concepts of the source ontologies.

Secondly, aggregation can be computed by a simple average where each of the matchers contributes equally to the final similarity value. An average value is calculated by taking the summation of the individual similarity values divided by the number of total matchers. For example, let there be *m* number of matchers. For a concept *con* in the ontologies Oa and Ob, each matcher produces its output as $\mathrm{Sim}_{\mathrm{mch}}$ that denotes the similarity value between the concept *con* in the ontologies Oa and Ob. Then the aggregated similarity value $\mathrm{Sim}_{\mathrm{con}}$ is the average value computed by summing the individual similarities $\mathrm{Sim}_{\mathrm{mch}}$ divided by the m (i.e., total number of matchers). For instance, in the example above m = 2 as we have two matchers (i.e., label and axiom). Aggregated similarity is calculated as (1 + 0.667)/2 = 0.833, with the following formula.

$$\mathrm{Sim}_{\mathrm{con}} = \frac{1}{\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}} (\mathrm{Sim}_{\mathrm{mch_i}})$$

Thirdly, aggregation can be computed as a weighted mean (or average) value on the basis of the weights (or importance) of the matchers that are given according to the human preference. We need these criteria because some matchers contribute more than others. For example, concept Label has the maximum weight as it shows the real significance of a concept. In the field of statistics, the weighted average is calculated with the following formula.

$$\text{Weighted Mean} = \frac{w_1x_1 + w_2x_2 + \cdots + w_nx_n}{w_1 + w_2 + \cdots + w_n} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

On the basis of this formula, weighted aggregated similarity value $\mathrm{Wg\_Sim}_{\mathrm{con}}$ is computed by multiplying the weight wg_{i} to the $\mathrm{Sim}_{\mathrm{mch}}$ value before taking their average. Let the user gives weight = 1 to the label matcher and weight = 0.8 to the axiom matcher, then the $\mathrm{Wg\_Sim}_{\mathrm{con}}$ is calculated as (1 * 1 + 0.8 * 0.667)/(1 + 0.8) = 0.852.

When the weights of label matcher and axiom matcher are same or equal to 1, then the weighted average produces the same value as simple average by (1 * 1 + 1 * 0.667)/(1 + 1) = 0.833. The following formula represents how $\mathrm{Wg\_Sim}_{\mathrm{con}}$ (weighted average) is computed when there are m number of matchers each with weight Wg.

$$\mathrm{Wg\_Sim_{con}} = \frac{\sum_{i=1}^{m} (wg_i * \mathrm{Sim}_{\mathrm{mch}_i})}{\sum_{i=1}^{m} wg_i}$$

5 Merging of axiomatic definitions of concepts

Merging of axiomatic definitions (or class expressions) is useful when we want to get an ontology in which concepts are expressed in the context with other concepts and properties. In addition, the merged ontology has the well-defined necessary and sufficient conditions over concepts from the source ontologies. This can be achieved by merging all the axiomatic definitions within the source ontologies together and by enriching the merged ontology with the axiomatic definitions of both the ontologies. But, the automatic merging of axiomatic definitions is much difficult and can be error-prone. It can become unsatisfiable, i.e., a merged axiom can lead to an unsatisfiable class definition to which no individual belongs. It can also become ambiguous as well giving no sense about the individual information or restrictions imposed. Therefore, we suggest that the final analysis should be made by the human Expert to meet the quality criteria if the automatic merged output will be used in a *critical* application.

There can be various scenarios regarding the merging of axiomatic definitions of mapped concepts. In the following subsections, we are discussing these scenarios individually (1) merging primitive and defined classes, and (2) merging two defined classes with their class expressions, with the piece of code that DKP-AOM exploits.

5.1 Merging a primitive class and a defined class from the source ontologies

Let us take a case of two classes, where Class clA is a primitive class and clB is a defined class by SomeValuesFrom Restriction. Our system follows the 7-Step algorithm (see Fig. 2) for their merging. For example, clA is a class MSCourse in O1 ontology and MSCourse is defined to be a class whose individuals are taught by some professor (i.e., MSCourse subclassof Taught_By some Professor), where Professor is described as a Person who teaches some Level_4_Course.

Our merge algorithm imports O1 ontology as a merged ontology O3 and then takes the snippets of axioms from O2 ontology to form the combined axiomatic definitions in O3. In the merged ontology O3, our system creates the restrictions of O2 when merging clA and clB candidates.The following piece of code (shown in Fig. 2) explains how our system works on merging a primitive and a defined class. The logic is that in the merged ontology, with OWLDataFactory mfactory, we create a new property newProp by which the restriction is applied in O2 with its domain and range concepts. Then, we get a class expression having the new restriction newRest and create a clA as a subclass of restriction class. This is explained in detail as seven steps mentioned in the Fig. 2.

5.2 Merging two defined classes from the source ontologies

There are various constructors for expression the real world objects into owl ontology concepts with class expressions. In general, we divide them into two categories based on the property match or mismatch. In case of a property mismatch, we have to create both the class expressions in the merged ontology by adding both the restrictions from the individual classes. But in the case of a property match, properties from the source ontologies themselves are merged and the union of filler classes should be generated in the merged ontology. Therefore, case 1 deals with the merging of two classes defined by the same restriction type

```
Input:  OWL class clA with no class expression
        OWL class clB with its class expression cl2
Function CreateRestrictionClass (OWLClass clA, ClassExpression cl2 , OWLClass clB)
{
        //clB definition is expressed by SomeValuesFrom Restriction cl2
        //example, clA= BSCourse, clB=BSCourse and its expression cl2: TaughtBy some Professor
        // where,Professor = Person and teaches some Level_4_Course.
        // in O3, our algorithm already imported O1, this means we already have clA = BSCourse in O3
        step 1.  Extract property (TaughtBy) and filler class (Professor) from class expression
                OWLObjectProperty newProp = getPropertyFrom( cl2);
                OWLClass Fcl2 = getFillerClass( cl2);

        Step 2. Create a new property as in the clB class expression in the merged ontology
                // mfactory is merged ontology factory that has already definition of clA
                OWLObjectProperty newProp = mfactory.getOWLObjectProperty( newProp );

    Step 3. Create domain and range of new property in the merged ontology
        OWLObjectPropertyDomainAxiom ax0 = mfactory.getOWLObjectPropertyDomainAxiom(newProp, clA);
        OWLObjectPropertyRangeAxiom ax1 = mfactory.getOWLObjectPropertyRangeAxiom(newProp, datarange);

        Step 4. Add new property domain and range in merged ontology
        AddAxiom addAx0 = new AddAxiom(mergedontology, ax0);
        AddAxiom addAx1 = new AddAxiom(mergedontology, ax1);

        step5. Making new restriction class expression in the merged ontology
        OWLClassExpression newRest = mfactory.getOWLObjectSomeValuesFrom(newProp, clas2);

        step6. Make clA class that is already in the merged ontology as a subclass restriction class
        OWLSubClassOfAxiom ax = mfactory.getOWLSubClassOfAxiom(clA, newRest );

        step 7. Add an axiom to the merged ontology and finally save it
        AddAxiom addAx = new AddAxiom(mergedOntology, ax);
        manager.saveOntology (mergedOntology);
  }
```

Fig. 2 Program creating a restriction class in the merged ontology

with two different properties and filler classes, and case 2 deal with the merging of two classes defined by the same restriction type with the same property and/or filler classes. The following subsections tackle these two cases individually.

5.2.1 Merging two classes defined by the same restriction type with two different properties and filler classes

Let us take a case of two classes (clA and clB) which are defined by the *SomeValuesFrom* Restriction but they use different object properties. For example, clA class MSCourse is defined to be a class to whom some AssistantProfessor assist (i.e., MSCourse subclassof *Assisted_By some AssistantProfessor*) where AssistantProfessor is described as a Person who teaches some Level_2_Course. In ontology O2, clB class MSCourse is defined to be a class to whom some FullProfessor teach (i.e., MSCourse subclassof *Taught_By some FullProfessor*) where FullProfessor is described as a Person who teaches some Level_4_Course. Our merge algorithm imports O1 ontology as a merged ontology O3 and then takes the snippets of axioms from O2 ontology to form the combined axiomatic definitions in O3. In the merged ontology O3, our system creates the restrictions of O1 and O2 when merging clA and clB candidate classes. The piece of code in Fig. 3 explains how our system works on merging two defined classes by class restrictions with different object properties. Our merge algorithm generates

```
Input: OWL class clA with its class expression cl1
       OWL class clB with its class expression cl2
Output: the merged ontology having two restriction classes with different properties.
Function MergeRestrictionClasses (ClassExpression cl1, OWLClass clA, ClassExpression cl2 , OWLClass clB)
{
/*clA and clB are classes with definitions which are expressed by the Restriction cl, and cl2
example, clA=BSCourse, cl1= TaughtBy some AssociateProfessor,
              where, AssociateProfessor = Person and teaches some level_2_course
clB=BSCourse, cl2=ConductedBy some FullProfessor
              where, FullProfessor = Person and teaches some level_4_course
note:   In O3, our algorithm already imported O1, this means we already have clA class
        clA with its class expression cl is already in the merged ontology
*/
   step 1. Extract a property (TaughtBy) and a filler class (AssociateProfessor) from the first class expression cl1
   step 2. Extract a property (ConductedBy) and a filler class (FullProfessor) from the second class expression cl2
   step 3. Creating new property as in clB class expression, with its domain and range in the merged ontology by
   merged factory (mfactory object)
   step 4. Add domain and range axioms in the merged ontology
   step 5. Make a new restriction class expression in the merged ontology
   step 6. Make clA class that is already in the merged ontology as a subclass of restriction class
   step 7. Add an axiom to the merged ontology and save it.
   }
```

Fig. 3 Program merging restriction classes with the different properties in the merged ontology

the merged class hierarchy by merging an input ontology O2 into the prioritized (or default) input ontology O1. After the generation of a class hierarchy, we need to merge axiomatic definitions from the source ontologies. Therefore, we need to execute this algorithm that consists of seven steps to get a merged axiomatic definition. Step by step working of the generation of merged axiomatic definition is shown in the Fig. 3. The main logic of the axiomatic definition generation is that in the merged ontology O3, we import clA with its original definition from O1. This means the merged ontology O3 has the axiomatic definition from the O1. As the properties involved in the class expressions are different, so they do not require to be merged. Therefore, we update the definition with the restriction from clB to get both the restrictions in the merged ontology O3 from the source ontologies. Below, this is explained in seven steps how DKP-AOM merge two classes defined by the same restriction type with two different properties and filler classes from the source ontologies.

5.2.2 Merging two classes defined by the same restriction type with the same property and/or filler classes

Another case can be possible when there are two class definitions which have the same or similar properties (that are mapped) involved, then the situation is different. In this case, we have to create a union of filler classes from the individual restriction classes, with the same property involved. Let us take a case of two classes (clA and clB) which are defined by the *SomeValuesFrom* Restriction, but comprise same object properties with the different filler classes. For example, clA class Scholar is defined to be a class whose individuals are registered in some conference (Scholar subclass-of *Registered_At some Conference*), and clB class Scholar is defined to be a class whose individuals are registered in some Tutorial (i.e., Scholar subclass-of *registered_At some Tutorial*). Our merge algorithm imports O1 ontology as a merged ontology O3 and then takes the snippets of axioms from O2 ontology to form the combined axiomatic definitions in O3. In the merged ontology O3, we have to combine the same properties as they are same/mapped, but, have to create a new filler class expression comprise of both concepts *Conference and Tutorial* [i.e., *Registered_At some (Conference or*

```
Input: OWL class clA with its class expression cl1
          OWL class clB with its class expression cl2
Output: The merged ontology having merged restriction classes with same properties.
Function MergeRestrictionClasses (ClassExpression cl1, OWLClass clA, ClassExpression cl2 , OWLClass clB)
{
          /*clA and clB are classes with definitions which are expressed by SomeValuesFrom //Restriction cl, and cl2
          example, clA=BSCourse, cl= TaughtBy some AssociateProfessor
                    clB=BSCourse, cl2=TaughtBy some FullProfessor
          note: In O3, our algorithm already imported O1, this means we already have clA class
          clA with its class expression cl is already in the merged ontology
          */

   step 1. Extract a property (TaughtBy) and a filler class (AssociateProfessor) from the first class expression cl1
   step 2. Extract a property (ConductedBy) and a filler class (FullProfessor) from the second class expression cl2
  step3. Delete clA subclass restriction from the merged ontology, it has already this restriction as in clA so delete
          itbefore updating the merged ontology with merged definition.
   Step4.  Create a new property as in clB class expression, with its domain and range
   Step5. Add axioms of domain and range in the merged ontology
   Step6. Merge filler as the union of concepts (AssociateProfessor or FullProfessor)
   Step 7. Make  a new restriction class expression in the merged ontology
   Step8. making clA class that is already in the merged ontology as a subclass of restriction class newRest
   Step 9. Add an axiom in the merged ontology and saving it

}//end some values from
```

Fig. 4 Program merging restriction classes with the same properties in the merged ontology

Tutorial)]. The piece of code in Fig. 4 explains how our system works on merging two defined classes by class restrictions with the same object properties. Our merge algorithm generates a merged class hierarchy by merging an input ontology O2 into the prioritized (or default) input ontology O1. After the generation of a class hierarchy, we need to merge axiomatic definitions from the source ontologies. Therefore, we need to execute this algorithm that consists of nine steps to get a merged axiomatic definition. The main logic is that as in the merged ontology, we have already imported clA with its original definition, therefore, first we have to delete its restriction subclassof axiom (its class expression and not class itself). Then, we have to create a new filler class expression with the filler classes from the individual restriction expression and then create a new restriction class with this new filler class expression. Finally, we have to update the merged ontology with this new restriction class as a subclass of clA.

6 Test case for mapping and merging of axiomatic definitions

This section presents a test case for the execution of our algorithms. This test case is divided into two sections. The first section discusses the mapping of axiomatic definitions and the second section demonstrates the merging of axiomatic definitions.

6.1 Mapping of OWL axiomatic definitions

Mapping axiomatic definitions is the most difficult task for merging concept definitions in the ontology merging process. We have created two versions of conference management system ontologies (extended crs_dr) in which they have different concept names with the semantically same (or overlapping) axiomatic definitions. This is obvious that concepts with different names are not identified by the string, linguistic or synonym strategies. Hence, the role of an axiomatic analysis of concepts becomes active. Otherwise, such concepts are not identified by the matching module of the merging system, and they create redundancy in the merged ontology. For example, consider the concepts in Table 2, *Article* in the ontology O1 and *Paper* in the ontology O2 are candidates for mapping and merging as they represent the semantically similar axiomatic definitions.

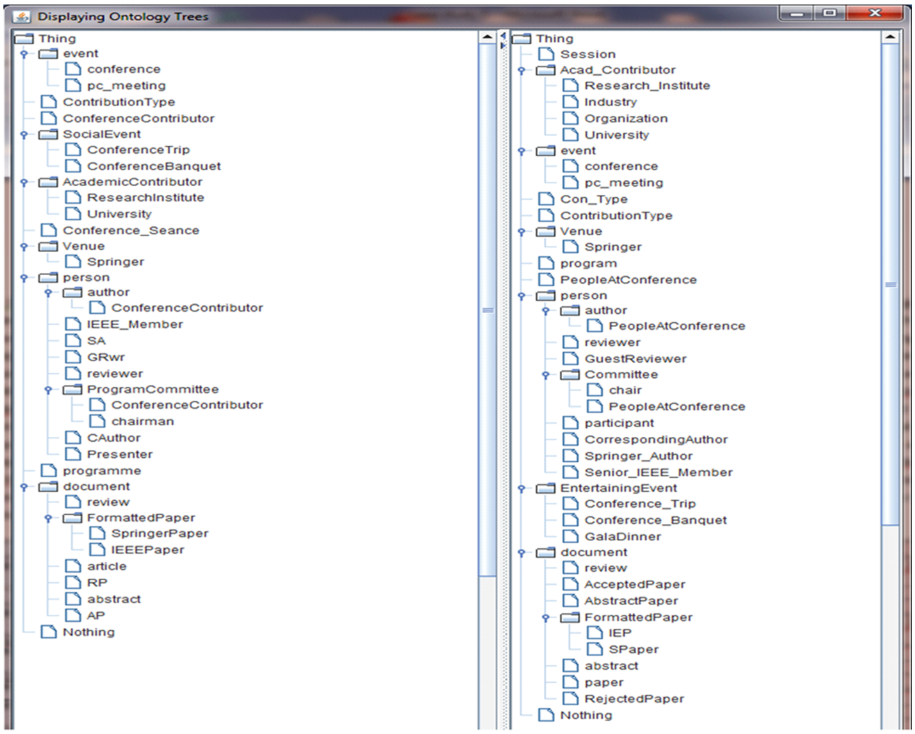
Another important point is to analyse, all the individual classes (named or anonymous made by different class constructors) within the axiomatic definition of a concept. It may partially match in some cases, hence proper weights or mismatches are computed. For example, *Paper* in the O2 ontology and *Abstract Paper* in the O1 with a definition [Abs_Paper subclassOf {document and (mustHave only abstract) and (writtenBy some author) and (has_author min 1)}] have partially same definitions, hence they require in depth analysis. Therefore, these should be properly mapped to each other, and similarity value with conflict is generated. The class hierarchies of these ontologies are shown in Fig. 5.

For the axiomatic definition analysis, the first step is to compute the basic concept mappings. Therefore, DKP-AOM first computed basic mappings (by the primary strategies, i.e., string, synonym, etc.), as these will be used in the axiomatic analysis of definitions/class expressions. The basic string-based and synonym-based mappings are shown in Fig. 6. Then, it analyses the class expressions and identify axiomatic mappings with the similarity value and/or conflict if any between the concepts of ontologies. Their axiomatic mappings produced by DKP-AOM are shown in Fig. 7. When the definitions are matched fully, their similarity value is calculated to be 1. But, when they have some conflict (e.g., difference in their restriction type), then their similarity is reduced as per values given/adjusted by the end-user. For example, *Some/All Values From* restriction creates a conflict of restriction type, so the similarity is reduced and a conflict is generated by the system. For instance, concept *accepted paper* in two ontologies is defined with the different terminology as: **AP** [document and (review_written some Chairman)] and *AcceptedPaper* [document and (review_written_by only Chair)]. These concepts are based on the initial mappings *(document, document)*, *(review_written, review_written_by)* and *(chairman, chair)*. But, they create a conflict in second anonymous restriction class between restriction types (some, i.e., 0..n) and (only/all, i.e., 1..n). Hence, in the running example, their similarity values are calculated to be 0.95.

Consider another example, where *contributionType* can be one of *{JournalInvitationPaper, FullPaper, abstractPaper}* in the ontology O1, but, it is one of {JournalInvitationPaper, abstractPaper} in the ontology O2. Their similarity by the axiomatic analysis only is calculated as 0.833 as per the rules defined in the initial chapters. However, this value indicates their axiomatic similarity, which is later on aggregated by their label similarity (i.e., 1.0) or with other criteria. When we executed these ontologies on *Prompt* merging system, it has only detected mappings based on the string similarity and produced the list of mappings (see Fig. 8) to the end-user for their manual merging. From analysing the list of mappings, we observe that 13 concept mappings are missing by Prompt (which can be detected by their axiomatic definitions by our system). The reason is that Prompt employs only string-based criteria to find candidate mappings and then suggest user to take decision for their merging. But, only the string-based criteria is not capable of finding candidates that have different terminology explicated by some abbreviation or compound words or some other terminology. Therefore, it is obvious that these 13 concepts from each ontologies compromise the recall value of matching result and create redundancy in the merged ontology. Our system DKP-AOM applies the axiomatic criteria defined along the proposed framework and detects these mappings. This results our system DKP-AOM to achieve higher recall value and conciseness in the merged ontology.

Table 2 Concepts with Axioms in the heterogeneous Conference Ontologies

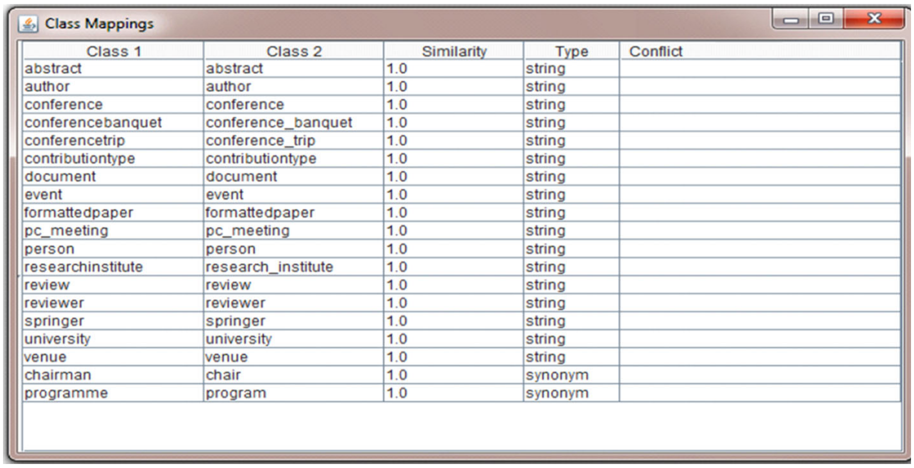
Concepts and Axioms in O1 ontology		Concepts and Axioms in O2 ontology	
Concept	Axiom	Concept	Axiom
Article	Document and (writtenBy some author) and (has_author min 1)	Paper	document and (written_By some author) and (hasAuthor min 1)
Abs_Paper	document and (mustHave only abstract) and (writtenBy some author) and (has_author min 1)	AbstractPaper	Document and (hasAuthor min 1) and (mustHave only abstract) and (written_by some author)
AP	Document and (Review_written some Chairman)	AcceptedPaper	Document and (review_written_by only Chair)
RP	Document and (RejectedBy only ProgramCommittee)	RejectedPaper	Document and (has_rejected some Committee)
IEEEPaper	FormattedPaper and (has_reviewer only reviewer)	IEP	FormattedPaper and (have_reviewer only reviewer)
SpringerPaper	FormattedPaper and (review_written some chairman)	SPaper	FormattedPaper and (review_written_by some chair)
Conference_Seance	Pc_meeting or Conference	Session	Pc_meeting or Conference
Conference Contributor	Author and ProgramCommittee	PeopleAt-Conference	Author and Committee
GRWr	Writes_review max 5	GuestReviewer	Write_reviews max 5
IEEE_Member	Person and (has_subscribed_At has "IEEE")	Senior_IEEE_Member	Person and (subscribed_At has "IEEE")
Presenter	Person and (has_published_At min 1)	Participant	Person and (published_At min 1)
SA	Person and (has_published_At some Springer)	Springer_Author	Person and (has_published_At some Springer)
Contribution Type	{JournalInvitationPaper, FullPaper, abstractPaper}	Contribution Type	{JournalInvitationPaper, abstractPaper}



```
Thing
  event
    conference
    pc_meeting
  ContributionType
  ConferenceContributor
  SocialEvent
    ConferenceTrip
    ConferenceBanquet
  AcademicContributor
    ResearchInstitute
    University
  Conference_Seance
  Venue
    Springer
  person
    author
      ConferenceContributor
    IEEE_Member
    SA
    GRwr
    reviewer
    ProgramCommittee
      ConferenceContributor
      chairman
    CAuthor
    Presenter
  programme
  document
    review
    FormattedPaper
      SpringerPaper
      IEEEPaper
    article
    RP
    abstract
    AP
  Nothing
```

```
Thing
  Session
  Acad_Contributor
    Research_Institute
    Industry
    Organization
    University
  event
    conference
    pc_meeting
  Con_Type
  ContributionType
  Venue
    Springer
  program
  PeopleAtConference
  person
    author
      PeopleAtConference
    reviewer
    GuestReviewer
    Committee
      chair
      PeopleAtConference
    participant
    CorrespondingAuthor
    Springer_Author
    Senior_IEEE_Member
  EntertainingEvent
    Conference_Trip
    Conference_Banquet
    GalaDinner
  document
    review
    AcceptedPaper
    AbstractPaper
    FormattedPaper
      IEP
      SPaper
    abstract
    paper
    RejectedPaper
  Nothing
```

Fig. 5 Input ontologies for axiomatic definitions analysis

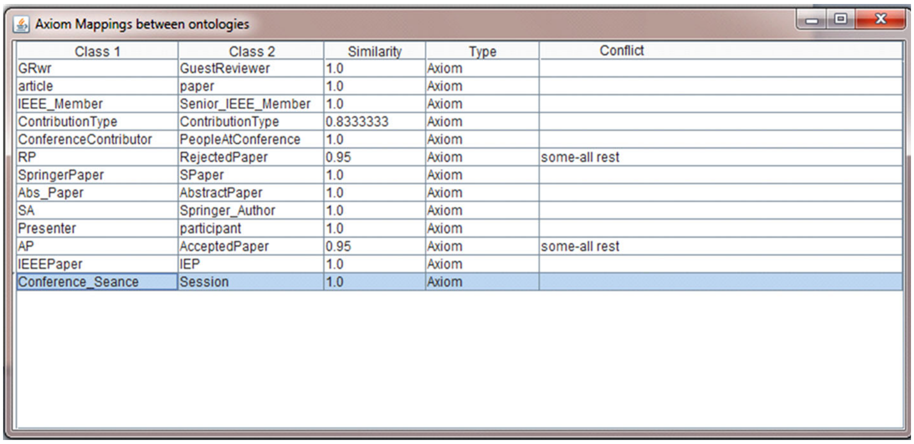


Class 1	Class 2	Similarity	Type	Conflict
abstract	abstract	1.0	string	
author	author	1.0	string	
conference	conference	1.0	string	
conferencebanquet	conference_banquet	1.0	string	
conferencetrip	conference_trip	1.0	string	
contributiontype	contributiontype	1.0	string	
document	document	1.0	string	
event	event	1.0	string	
formattedpaper	formattedpaper	1.0	string	
pc_meeting	pc_meeting	1.0	string	
person	person	1.0	string	
researchinstitute	research_institute	1.0	string	
review	review	1.0	string	
reviewer	reviewer	1.0	string	
springer	springer	1.0	string	
university	university	1.0	string	
venue	venue	1.0	string	
chairman	chair	1.0	synonym	
programme	program	1.0	synonym	

Fig. 6 Basic mappings of concepts before axiomatic mappings

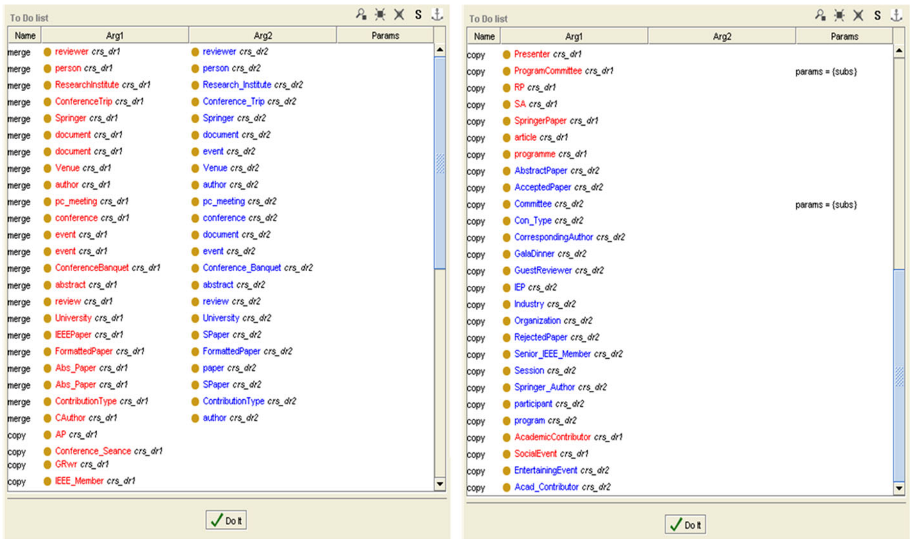
6.2 Merging of OWL axiomatic definitions

The aim of this test case is to get an enriched merged ontology by merging axiomatic definitions (or class expressions) from the source ontologies. For this test case, we have *created*



Class 1	Class 2	Similarity	Type	Conflict
GRwr	GuestReviewer	1.0	Axiom	
article	paper	1.0	Axiom	
IEEE_Member	Senior_IEEE_Member	1.0	Axiom	
ContributionType	ContributionType	0.8333333	Axiom	
ConferenceContributor	PeopleAtConference	1.0	Axiom	
RP	RejectedPaper	0.95	Axiom	some-all rest
SpringerPaper	SPaper	1.0	Axiom	
Abs_Paper	AbstractPaper	1.0	Axiom	
SA	Springer_Author	1.0	Axiom	
Presenter	participant	1.0	Axiom	
AP	AcceptedPaper	0.95	Axiom	some-all rest
IEEEPaper	IEP	1.0	Axiom	
Conference_Seance	Session	1.0	Axiom	

Fig. 7 Axiomatic mappings of concepts after their class expression analysis



Name	Arg1	Arg2	Params
merge	reviewer *crs_dr1*	reviewer *crs_dr2*	
merge	person *crs_dr1*	person *crs_dr2*	
merge	ResearchInstitute *crs_dr1*	Research_Institute *crs_dr2*	
merge	ConferenceTrip *crs_dr1*	Conference_Trip *crs_dr2*	
merge	Springer *crs_dr1*	Springer *crs_dr2*	
merge	document *crs_dr1*	document *crs_dr2*	
merge	document *crs_dr1*	event *crs_dr2*	
merge	Venue *crs_dr1*	Venue *crs_dr2*	
merge	author *crs_dr1*	author *crs_dr2*	
merge	pc_meeting *crs_dr1*	pc_meeting *crs_dr2*	
merge	conference *crs_dr1*	conference *crs_dr2*	
merge	event *crs_dr1*	document *crs_dr2*	
merge	event *crs_dr1*	event *crs_dr2*	
merge	ConferenceBanquet *crs_dr1*	Conference_Banquet *crs_dr2*	
merge	abstract *crs_dr1*	abstract *crs_dr2*	
merge	review *crs_dr1*	review *crs_dr2*	
merge	University *crs_dr1*	University *crs_dr2*	
merge	IEEEPaper *crs_dr1*	SPaper *crs_dr2*	
merge	FormattedPaper *crs_dr1*	FormattedPaper *crs_dr2*	
merge	Abs_Paper *crs_dr1*	paper *crs_dr2*	
merge	Abs_Paper *crs_dr1*	SPaper *crs_dr2*	
merge	ContributionType *crs_dr1*	ContributionType *crs_dr2*	
merge	CAuthor *crs_dr1*	author *crs_dr2*	
copy	AP *crs_dr1*		
copy	Conference_Seance *crs_dr1*		
copy	GRwr *crs_dr1*		
copy	IEEE_Member *crs_dr1*		

Name	Arg1	Arg2	Params
copy	Presenter *crs_dr1*		
copy	ProgramCommittee *crs_dr1*		params = {subs}
copy	RP *crs_dr1*		
copy	SA *crs_dr1*		
copy	SpringerPaper *crs_dr1*		
copy	article *crs_dr1*		
copy	programme *crs_dr1*		
copy	AbstractPaper *crs_dr2*		
copy	AcceptedPaper *crs_dr2*		
copy	Committee *crs_dr2*		params = {subs}
copy	Con_Type *crs_dr2*		
copy	CorrespondingAuthor *crs_dr2*		
copy	GalaDinner *crs_dr2*		
copy	GuestReviewer *crs_dr2*		
copy	IEP *crs_dr2*		
copy	Industry *crs_dr2*		
copy	Organization *crs_dr2*		
copy	RejectedPaper *crs_dr2*		
copy	Senior_IEEE_Member *crs_dr2*		
copy	Session *crs_dr2*		
copy	Springer_Author *crs_dr2*		
copy	participant *crs_dr2*		
copy	program *crs_dr2*		
copy	AcademicContributor *crs_dr1*		
copy	SocialEvent *crs_dr1*		
copy	EntertainingEvent *crs_dr2*		
copy	Acad_Contributor *crs_dr2*		

Fig. 8 List of mappings by prompt merging system (*many of them are incorrect and not detected as compared to our system see Figs. 6 and 7)

two versions of conference ontologies, which have almost the same concepts, but with different axiomatic definitions (as our aim is to analyse the merging of axiomatic definitions). There are many cases for the merging of axiomatic definitions. Therefore, in these ontologies many of the concepts are left primitive and others are defined with the axiomatic definitions. Table 3 represents concepts from these ontologies and their axiomatic definitions. The last column shows how DKP-AOM generated the merged axioms based on the axioms from the source ontologies. Some of the concepts in the ontology O1 are primitive without any class expression. But, in the ontology O2, they are defined with class expressions defining their means in terms of other concepts. For example, *ExternalReviewer* is a primitive concept in an ontology O1 and is defined as a subclass of {*Reviews some ConferencePaper*} in an ontology

O2. During the merging of such concepts, DKP-AOM has merged the concepts and created a class expression in the merged ontology as it is defined in an ontology O2.

In another case, when both the concepts are defined by the class expressions, we have to merge their class expressions to get the enriched class expression in the merged ontology. By merging these concepts, we get an ontology that provides much more vocabulary supported with their individual axiomatic definitions. For example, consider a simple situation where a concept *PeopleAtConference* is the union of Person, Author or Admin_Staff concepts in the ontology O1. But, in the ontology O2, it is defined as a union of PCMember or Reviewer or Presenter or Admin_Staff. Our system makes a new definition by merging both the local definitions by making the union of both the set of concepts. Table 3 shows the axioms of concepts from the O1 and O2 ontologies and the corresponding axioms generated in the merged ontology O3 by our DKP-AOM system.

From this test case, we conclude our ontology merging algorithms presented in previous sections perform very well and facilitate the merging process when one wants to get an ontology with the better quality. Each of the axiomatic definitions from the source ontologies are matched together, merging is performed on them, and combined rich axioms are added in the merged ontology. More details about the input and output ontologies and other aspects of DKP-AOM tool and its implementation can be found in Fahad (2015). Merging of axiomatic definitions really achieves richer merged ontology which captures sufficient definitions from the source ontologies. But, it is possible that automatic merging creates an unsatisfiable definition of a concept while merging of boolean combinations in the class expression of concepts. Therefore, we suggest that axioms in the merged ontology should be analysed after the automatic merging by DKP-AOM system, because the process of their conflict-free integration is highly error-prone. Especially when the resultant automatic merged ontology will be used in some critical application.

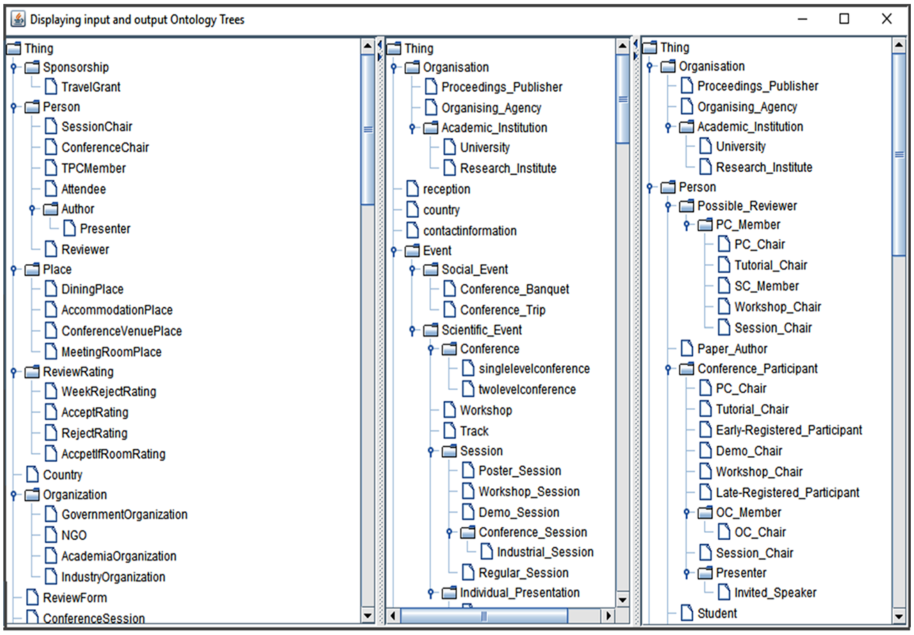
Conference ontologies provided by OAEI are the best for testing our contribution as they are well equipped with DL axioms. Consider Ekaw and Edas ontologies from the conference data set. Ekaw ontology has 74 classes having DL expressivity in SHIN and Edas ontology has 104 classes having DL expressivity ALCOIN(D). Figure 9 illustrates the merging of Ekaw and Edas ontologies. The first version of our tool with the implementation of algorithms proposed in this paper is launched and can be downloaded and tested from here.[1] The mapping system is separated from the merging system, and can be downloaded according to needs. For the merging of ontologies, use the same command of seals platform with –o following three paths, two for source ontologies and one for the output merged ontology. As a result of this command, a list of ontology mappings and a resultant merged ontology are produced.

As compared to two available *(regardless there are many approaches without prototype)* ontology merging tools, *Prompt* (Bock et al. 0000) and *Atom* (Raunich and Rahm 2012), this aspect of the automatic axiomatic definition merging is novel in our *DKP-AOM*. *ATOM*, regardless of its automatic nature for merging ontologies, does not provide this feature and is helpful for merging only owl-lite ontologies. We have incorporated the complexity of OWL2 ontologies that represent the real world objects by defining class expressions. At this moment, our system DKP-AOM is capable of automatically merging classes having class definitions expressed by OWL2 primitives (Morphadorner 2010), such as; *ObjectSomeValuesFrom, ObjectAllValueFrom, ObjectHasValue, ObjectMinCardinality, ObjectMaxCardinality, ObjectUnionOf, ObjectIntersectionOf, DataSomeValuesFrom, DataAllValueFrom, DataHasValue, DataMinCardinality, and DataMaxCardinality*.

[1] DKP-AOM: ontology merging tool, https://sites.google.com/site/mhdfahad/plugins.

Table 3 Same concept with axiomatic definitions in input and output ontologies

Concept Label	Axiom in O1	Axiom in O2	Merged Axiom in O3
Person	acceptPaper Some Conference_Paper	ReadsSome Conference_Paper	acceptPaper Some Conference_Paper reads Some Conference_Paper
Conference_Paper	ReviewedBy Some PCMember	ReviewedBy Some Reviewer	ReviewedBy Some (PCMember and Reviewer)
AbtractPaper	markedBy only GuestReviewer	ReviewedBy only PCMember	markedBy only GuestReviewer ReviewedBy only PCMember
RejectedPaper	RejectedBy only GuestReviewer	RejectedBy only Chair	RejectedBy only (GuestReviewer and Chair)
Author	PeopleAtConference and Person	PeopleAtConference and Presenter	PeopleAtConference and Person and Presenter
PeopleAt Conference	Person or Author or Admin_Staff	PCMember or Reviewer or Admin_Staff orPresenter	Person or Author or PCMember or Revieweror Presenter or Admin_Staff
GoodPaper	isAcceptedAs hasValue wellWritten	isAcceptedAs hasValue fullPaper	isAcceptedAs hasValue wellWritten isAcceptedAs hasValue fullPaper
BestPaper	hasRank hasValue 10	hasStatus hasValue Excellent	hasRank hasValue 10 hasStatus hasValue Excellent
Contribution	has_Keyword min 3	has_CorrespondingAuthor min 1	has_Keyword min 3 has_CorrespondingAuthor min 1
AwardWinner_Article	–	GoodPaper and BestPaper	GoodPaper and BestPaper
External Reviewer	–	Reviews some ConferencePaper	Reviews some ConferencePaper
Presentation Duration	–	limitedTo hasValue 20	limitedTo hasValue 20
Submission	–	Conference_Paper and Tutorial	Conference_Paper and Tutorial
Scholar	–	registeredAt only Organization	registeredAt only Organization



Displaying input and output Ontology Trees

- Thing
  - Sponsorship
    - TravelGrant
  - Person
    - SessionChair
    - ConferenceChair
    - TPCMember
    - Attendee
    - Author
      - Presenter
    - Reviewer
  - Place
    - DiningPlace
    - AccommodationPlace
    - ConferenceVenuePlace
    - MeetingRoomPlace
  - ReviewRating
    - WeekRejectRating
    - AcceptRating
    - RejectRating
    - AccpetIfRoomRating
  - Country
  - Organization
    - GovernmentOrganization
    - NGO
    - AcademiaOrganization
    - IndustryOrganization
  - ReviewForm
  - ConferenceSession

- Thing
  - Organisation
    - Proceedings_Publisher
    - Organising_Agency
    - Academic_Institution
      - University
      - Research_Institute
  - reception
  - country
  - contactinformation
  - Event
    - Social_Event
      - Conference_Banquet
      - Conference_Trip
    - Scientific_Event
      - Conference
        - singlelevelconference
        - twolevelconference
      - Workshop
      - Track
      - Session
        - Poster_Session
        - Workshop_Session
        - Demo_Session
        - Conference_Session
          - Industrial_Session
        - Regular_Session
      - Individual_Presentation

- Thing
  - Organisation
    - Proceedings_Publisher
    - Organising_Agency
    - Academic_Institution
      - University
      - Research_Institute
  - Person
    - Possible_Reviewer
      - PC_Member
        - PC_Chair
        - Tutorial_Chair
        - SC_Member
        - Workshop_Chair
        - Session_Chair
    - Paper_Author
    - Conference_Participant
      - PC_Chair
      - Tutorial_Chair
      - Early-Registered_Participant
      - Demo_Chair
      - Workshop_Chair
      - Late-Registered_Participant
      - OC_Member
        - OC_Chair
      - Session_Chair
      - Presenter
        - Invited_Speaker
    - Student

Fig. 9 Merging of Ekaw and Edas conference ontologies

7 Comparison with ontology matchers from OAEI 2012

This section presents an evaluation of our mapping module of DKP-AOM. Due to unavailability of merging systems, we decided to compare analytical results with some of the mapping systems. We present major results with the Group 1 (good performance matchers) and Group 2 (worse performance matcher) according to final results of OAEI 2012, more details of these tests can be found in Fahad (2015). The top three good performance matchers are YAM++, LogMap and CODI. The worse performance matchers are ServOMap, ServoMapLt, MapSSS, and AUTOMSv2. The results of OAEI 2012 are publically available at http://oaei.ontologymatching.org/2012/conference/eval.html. We extracted the result of these matchers from their published results and compared it with our system DKP-AOM.

Ngo and Ellahsene developed *YAM++* which employs a multi-strategy based approach for ontology matching task (Ngo and Bellahsene 2012a, b). It exploits information retrieval techniques for the identification of mappings, and also a machine learning approach to identify correspondences between entities of source ontologies when training dataset is available. It is different from others as it is also capable to handle multi-lingual ontologies. Another tool named *LogMap* is developed by Jiménez-Ruiz and Cuenca-Grau (2011); Jiménez-Ruiz et al. (2012) which is based on the reasoning and diagnosis capabilities. They claimed that their tool can efficiently deal with the bio-medical ontologies as well. Huber et al. (2011) developed another tool named *CODI* based on probabilistic-logical technique for the ontology mapping between individuals, concepts and properties. They performed corresponding combinatorial optimization problems for the identification of alignments. Both tools, LogMap and CODI, resemble with the pre-integration phase of our tool *DKP-AOM* (Fahad et al. 2011, 2012) because of the consistency and satisfiability analysis during the mapping of concepts between ontologies. But, we also took the initiative of merging the consistent candidate mappings to produce a new merged ontology, which is considered the main contribution of this paper. Therefore, this section is mainly discussing the comparative analysis between our system and other similar contributions.

Ba and Diallo (2012, 2013) developed *ServOMap* and *ServOMapLt* for the large-scale biomedical ontology matching for the integration of data among biomedical applications. They used an Ontology Server (ServO) and employed information retrieval techniques for finding the correspondences between source ontologies. Their designed server, ServO, serves as a semantic index that could be maintained in the memory which stores knowledge that could be used later for the identification of mappings between biomedical ontologies. Cheatham and HitzlerP, (2013) developed *MapSSS* for the *String Similarity Metrics* for the Ontology Alignment task. They evaluated a wide range of string similarity metrics and also employed string pre-processing strategies for finding the correspondences between ontologies. It resembles our tool, as we also used *MorphAdorner* for the pre-processing of ontology terminologies. They presented that if optimal string similarity metrics are used, then those alone can produce mappings that are competitive with the state-of-the-art in ontology alignment systems. Kotis et al. (2012) developed AUTOMSv2 by using open source Java Alignment API by INRIA. They focused on the aggregation by using lexical, structural, semantic and instance matching strategies with different aggregation operators.

Test Data Ontologies from Conference Track at OAEI 2012.

We choose many conference ontologies from OAEI 2012,[2] which can be downloaded from the direct link.[3] The results of all the matchers that participated in OAEI 2012 can be downloaded from this link.[4]

Experimental setting of DKP-AOM.

We tested DKP-AOM with two configurations on the test dataset 1, 2 and 3 provided below.

- First (denoted as DKP-AOM (v1)), string matcher (with threshold value = 1), synonym matcher, inheritance matcher, Property matcher (without any restriction on domain concept).
- Second (denoted as DKP-AOM (v2)), string matcher (with threshold value = 1), synonym matcher, inheritance matcher, Property matcher (with restriction on domain concept match, i.e., two properties are mapped only if their domain concepts are mapped).

7.1 Comparison of DKP-AOM with Group 1 and 2 matchers of OAEI 2012

The goal of this section is to compare DKP-AOM on a standard ontology dataset (cmt and conference) from the OAEI 2012 and present analytical results. The details are presented below.

DataSet 1: Ontologies (Cmt and Conference)

Matchers Comparative test: We selected the following matchers for the comparison.

Matchers in Group-1: YAM++, LogMap, CODI
Matcher in Group-2: servOMap, servOMApL, MapSSS, AUTOSMv2

Analysis. Based on the results of Table 4, DKP (v1) has detected 12 accurate mappings for the merging of conference ontologies. It also has missed 3 mappings, which according to human

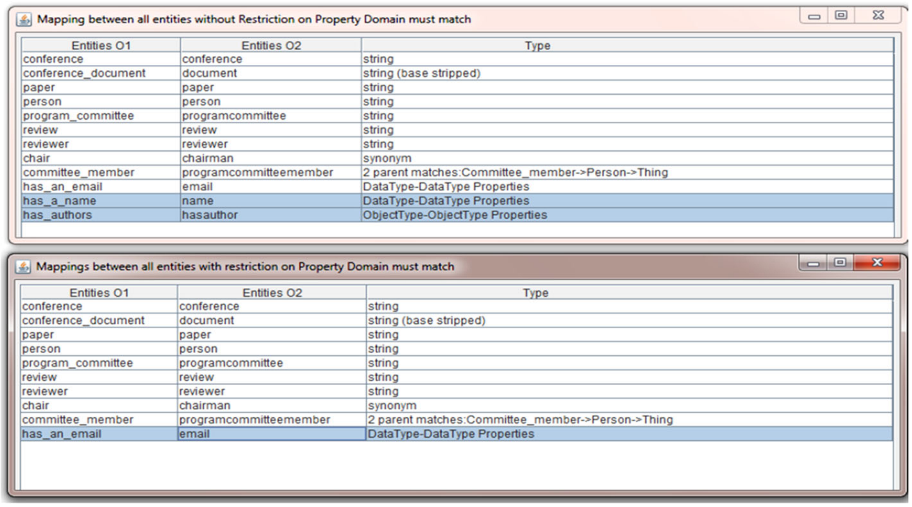
[2] http://oaei.ontologymatching.org/2012/conference/index.html.

[3] http://oaei.ontologymatching.org/2012/conference/data/conference.zip.

[4] http://oaei.ontologymatching.org/2012/conference/data/conference-alignments.zip.

Table 4 Comparison between DKP-AOM and Group-2 matchers of OAEI 2012

	Missed	Accurate	Inaccurate
Ontologies: Cmt and Conference			
servOMap	7	8	0
servOMapL	11	4	0
MapSSS	8	7	1
AUTOSMv2	10	5	0
DKP-AOM(v1)	3	12	0
DKP-AOM(v2)	4	11	0


Mapping between all entities without Restriction on Property Domain must match

Entities O1	Entities O2	Type
conference	conference	string
conference_document	document	string (base stripped)
paper	paper	string
person	person	string
program_committee	programcommittee	string
review	review	string
reviewer	reviewer	string
chair	chairman	synonym
committee_member	programcommitteemember	2 parent matches:Committee_member->Person->Thing
has_an_email	email	DataType-DataType Properties
has_a_name	name	DataType-DataType Properties
has_authors	hasauthor	ObjectType-ObjectType Properties

Mappings between all entities with restriction on Property Domain must match

Entities O1	Entities O2	Type
conference	conference	string
conference_document	document	string (base stripped)
paper	paper	string
person	person	string
program_committee	programcommittee	string
review	review	string
reviewer	reviewer	string
chair	chairman	synonym
committee_member	programcommitteemember	2 parent matches:Committee_member->Person->Thing
has_an_email	email	DataType-DataType Properties

Fig. 10 Mappings produced by DKP-AOM on Cmt and Conference ontologies

should be mapped and merged in the merged ontology. DKP-AOM (v2) has rejected property (has_a_name, name) due to domain mismatch as we selected this criterion (see above mode of execution in an experimental setting). DKP-AOM (v1) has performed well on the conference domain ontologies as compared to all the matchers. Precisely, it has outperformed as compared to Group 2 matchers which are considered as worse matchers by OAEI 2012, as these matchers has detected a very low number of accurate mappings and also missed a lot. Figure 10 shows the list of mappings produces by the DKP-AOM with configuration (v1 and v2).

As compared to Group 1 (see Table 5), it has performed very well as compared to CODI and LogMap, and produced almost the same results with the best matcher YAM++ (difference of only one). Therefore, we ignore the Group 2 matchers for the next comparison and decided to match the results of DKP-AOM only with Group 1 matchers. By manual inspection, it has missed mappings between (co-author, contribution_co-author), (paper, abstractpaper), (siteURL, has_a_URL) which are candidates for the merging according to human expert. These mappings can be detected, if we lower the threshold value or check the containment of one label into another, for instance (paper, abstractpaper). But, this compromises the precision value by detecting inaccurate mappings as well. For example, in conference domain ontologies, by such strategy by lowering the threshold value can detect (paper, abstractpaper) mapping, but it also detects mappings between (paper, invitedpaper), (paper, journal_Paper), (Paper, accepted-Paper) which are not candidates for merging.

Table 5 Comparison between DKP-AOM and Group-1 matchers of OAEI 2012

	Missed	Accurate	Inaccurate
Ontologies: Cmt and Conference			
CODI	6	9	1
LogMap	6	9	2
YAM++	4	11	1
DKP-AOM(v1)	3	12	0
DKP-AOM(v2)	4	11	0

Table 6 Comparison of DKP-AOM and Group 1 Matchers on (Edas, Sigkdd) and (cmt, iasted) ontologies

Conference ontologies: Edas and Sigkdd				cmt+iasted		
	Missed	Accurate	Inaccurate	Missed	Accurate	Inaccurate
CODI	1	8	0	0	5	0
LogMap	2	7	0	0	5	0
YAM++	1	8	0	0	5	0
DKP-AOM(v1)	0	9	0	0	5	0
DKP-AOM(v2)	2	7	0	0	5	0

7.1.1 Comparison of DKP-AOM with Group 1 matchers of OAEI 2012

This goal of this section is to compare DKP-AOM on standard ontology datasets from the OAEI 2012 and present our analytical results. The details are presented below.

DataSet 2: Ontologies (edas and sigkdd) and (cmt and iasted)

Matchers in Group-2: YAM++, LogMap, CODI

Analysis. On the basis of Table 6 for ontologies cmt and iasted, all the systems produced the similar results. All systems have detected string-based concept mappings. But, for ontologies edas and sigkdd, they produced interesting results. Figure 11 shows the list of mappings produced by DKP-AOM with configurations v1 and v2. We observe that DKP-AOM (v1) has detected 9 accurate mappings and missed none. But, YAM++ has also missed 1 mapping. From the analysis of dataset 1 and 2, we observed that our system DKP-AOM also has a potential to give good performance, so we decided to match it only with the best matcher YAM++.

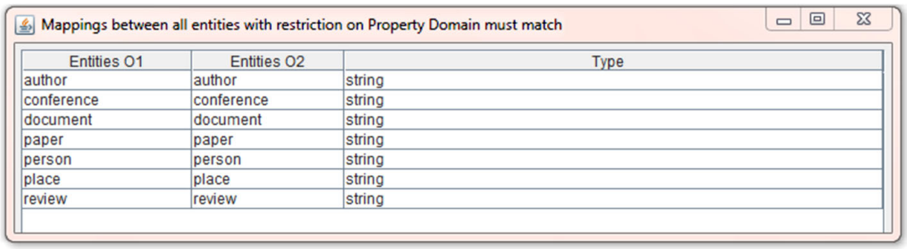
7.1.2 Comparison of DKP-AOM with YAM++ (best matcher of OAEI 2012)

The goal of this section is to compare DKP-AOM with YAM++ on standard ontology datasets from the OAEI 2012 and present our analytical results. The details are presented below.

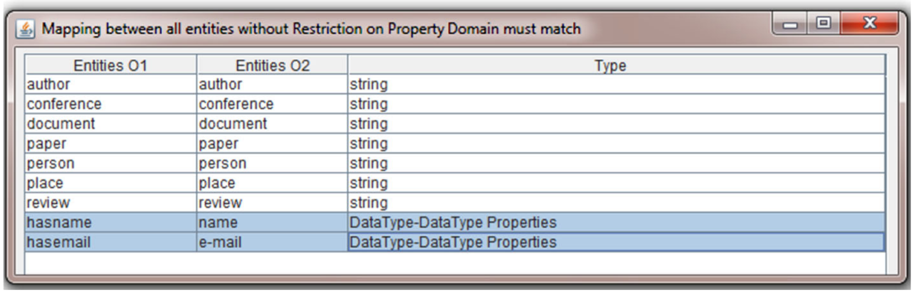
DataSet 3: Ontologies Conference, confOf, Sigkdd, and iasted.

Matchers: YAM++, DKP-AOM (v1 and v2)

Analysis. Based on the results from Table 7, we observe that DKP-AOM and YAM++ has demonstrated almost the same performance. DKP-AOM (v2) with restriction on the data


Mappings between all entities with restriction on Property Domain must match

Entities O1	Entities O2	Type
author	author	string
conference	conference	string
document	document	string
paper	paper	string
person	person	string
place	place	string
review	review	string


Mapping between all entities without Restriction on Property Domain must match

Entities O1	Entities O2	Type
author	author	string
conference	conference	string
document	document	string
paper	paper	string
person	person	string
place	place	string
review	review	string
hasname	name	DataType-DataType Properties
hasemail	e-mail	DataType-DataType Properties

Fig. 11 Mappings produced by DKP-AOM on Edas and Sigkdd ontologies

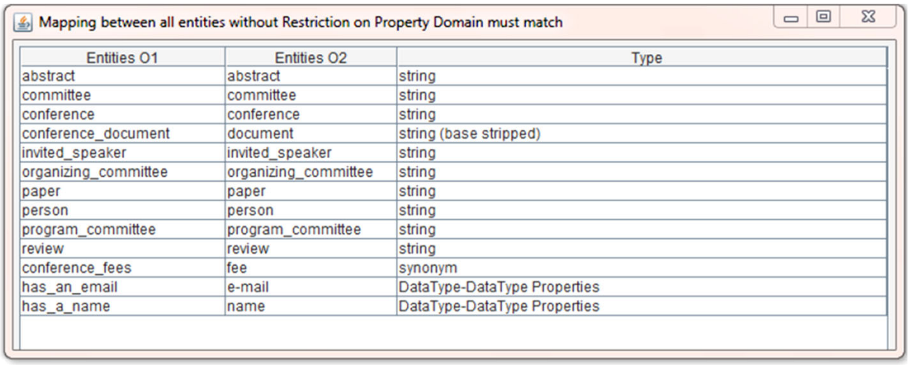
Table 7 Comparison of DKP-AOM with YAM++

Ontologies	conference-confOf			conference-Sigkdd			conference-iasted		
System	Missed	Accurate	Inaccurate	Missed	Accurate	Inaccurate	Missed	Accurate	Inaccurate
YAM++	1	12	0	1	12	0	1	6	0
DKP-AOM(v1)	0	13	0	0	13	0	0	7	1
DKP-AOM(v2)	1	12	0	1	12	0	1	6	1

property domain has shown very similar results, and DKP-AOM (v1) without any restriction on data property has shown a little better performance in some cases. Figure 12 shows the list of mappings produced by DKP-AOM with configurations v1 and v2. By the manual inspection, we observe that DKP-AOM (v1 and v2) detected one inaccurate mapping between concepts (Poster and Card) by synonym technique. But, in fact these are different concepts (i.e., Poster is a kind of contribution without oral presentation and Card can be a credit card for payment). These mappings can be ignored or rejected on the basis of disjointness between these concepts. But, these ontologies have not incorporated disjoint axioms between them. Hence, our system has not detected and considered as synonym based on the WordNet lexical database. Therefore, one of our future directions is to automatically learn disjointness (if it is not provided in source ontologies) between the concepts so that we improve more precision and recall values of our system.

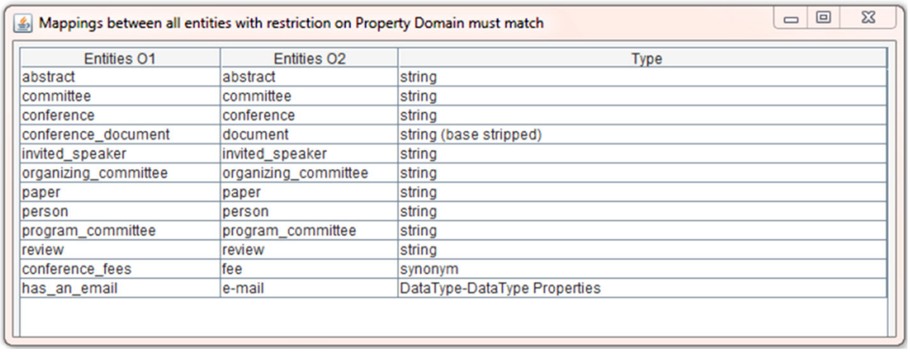
In the research literature, we have four metrics for measuring different perspectives of an automatic system. These are *Precision*, *Recall*, *F-measure* and *OverAll*, calculated as below.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{F-measure} = \frac{2{*}\text{Precision}{*}\text{Recall}}{\text{Precision} + \text{Recall}}$$



Mapping between all entities without Restriction on Property Domain must match

Entities O1	Entities O2	Type
abstract	abstract	string
committee	committee	string
conference	conference	string
conference_document	document	string (base stripped)
invited_speaker	invited_speaker	string
organizing_committee	organizing_committee	string
paper	paper	string
person	person	string
program_committee	program_committee	string
review	review	string
conference_fees	fee	synonym
has_an_email	e-mail	DataType-DataType Properties
has_a_name	name	DataType-DataType Properties



Mappings between all entities with restriction on Property Domain must match

Entities O1	Entities O2	Type
abstract	abstract	string
committee	committee	string
conference	conference	string
conference_document	document	string (base stripped)
invited_speaker	invited_speaker	string
organizing_committee	organizing_committee	string
paper	paper	string
person	person	string
program_committee	program_committee	string
review	review	string
conference_fees	fee	synonym
has_an_email	e-mail	DataType-DataType Properties

Fig. 12 Mappings produced by DKP-AOM on conference and Sigkdd ontologies

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{OverAll} = \text{Recall}^{*}\left(2 - \frac{1}{\text{Precision}}\right)$$

For this test, we need to compare Precision and Recall metrics to evaluate the accuracy of our system as compared to YAM++. From the Table 7, we aggregated the values of True Positive (TP), True Negative (TN) and False Positive (FP) as shown in Table 8. We analyse that the values are almost same generated by DKP-AOM and YAM++. DKP-AOM (v1) has shown a little better precision values than the other. Especially, DKP-AOM(v1) has gained more recall than others. It means that it has missed less values as compared to DKP-AOM(v2) and YAM++. Based on these analytical results, we can conclude that DKP-AOM has embedded different matching and NLT techniques to detect accurate mappings. It has also embedded with different strategies to avoid inaccurate mappings between concepts of conference ontologies. However, strict criteria to achieve precision of result lead DKP-AOM to miss many mappings (such as in the case of cmt and conference ontologies).

8 DKP-AOM results for OAEI 2015

The Ontology Alignment Evaluation Initiative is a coordinated international initiative assessing strengths and weaknesses of alignment/matching systems. Due to unavailability of merging systems, we are unable to conduct an experimental comparative analysis between

Table 8 Comparison of DKP-AOM and YAM++

	Missed(TN)	Accurate(TP)	Inaccurate(FP)	Precision	Recall	F-meaure	OverAll
Ontologies from DataSet 1, 2 and 3							
YAM++	8	54	1	0.981	0.870	0.923	0.854
DKP-AOM(v1)	3	59	1	0.983	0.951	0.967	0.935
DKP-AOM(v2)	9	53	1	0.981	0.854	0.913	0.838

them. Therefore, we participated in the OAEI 2015,[5] in order to show the efficiency and effectiveness of our system and here we discuss our successful participation in the Conference, OA4QA and Anatomy tracks. The results are very encouraging provided by the OAEI 2015 campaign as our system is acceptable and comparable with other participants (Fahad 2015). In OA4QA track, DKP-AOM out-performed in the evaluation. Precision and recall has calculated with respect to the ability of the generated alignments to answer a set of queries in an ontology-based data access scenario where several ontologies exist on three different data sets (i.e., RA1, RAR1, RA2 see official site for details). DKP-AOM is among four other ontology matchers (AML, LogMap, LogMapC and XMap) whose alignments allowed answering all the queries of the evaluation. The best global results have been achieved for violations queries that have been correctly covered w.r.t. RA1. Notably, DKP-AOM achieved an impressive f-measure of 0.999 w.r.t. RAR1, showing an effective handling of logical violations. DKP-AOM also participated in Anatomy track where task is placed in a domain where we find large, carefully designed ontologies that are described in technical terms. The anatomy real world case is about matching the adult mouse anatomy (2744 classes) and the NCI Thesaurus (3304 classes) describing the human anatomy. In the anatomy track, it has produced alignments within an allocated time and appeared in the list of *seven* systems which produce only coherent results. It has also generated only trivial correspondences.

In the scope of this paper, we are elaborating results of our system on the Conference domain ontologies because these ontologies provided for OEAI 2015 are suitable for showing the significance of our system. The goal of conference track is to find alignments among 16 ontologies relatively smaller in size (between 14 and 140 entities) but rich in semantic heterogeneities about the conference organization domain. These ontologies are rich in DL Axiomatic definitions which provide excellent opportunity to test and validate our contribution. As a result of OAEI, Alignments are evaluated automatically against reference alignments. Therefore, it is very interesting to measure the Precision, Recall and F-measure of our system on ontologies rich in OWL DL axioms of various kinds, and also does a comparison between existing systems to see their performance on real world datasets (see detailed results[6]).

8.1 Evaluation based on sharp reference alignments

The resultant match quality was evaluated against the original (ra1) as well as entailed reference alignment (ra2) and violation free version of reference alignment (rar2). The results are categorized in three groups. DKP-AOM is included in the Group 1 matchers. Group 1 consists of matchers (AML, Mamba, LogMap-C, LogMap, XMAP, GMap, DKP-AOM and

5 http://oaei.ontologymatching.org/2015/results/index.html.

6 http://oaei.ontologymatching.org/2015/conference/eval.html.

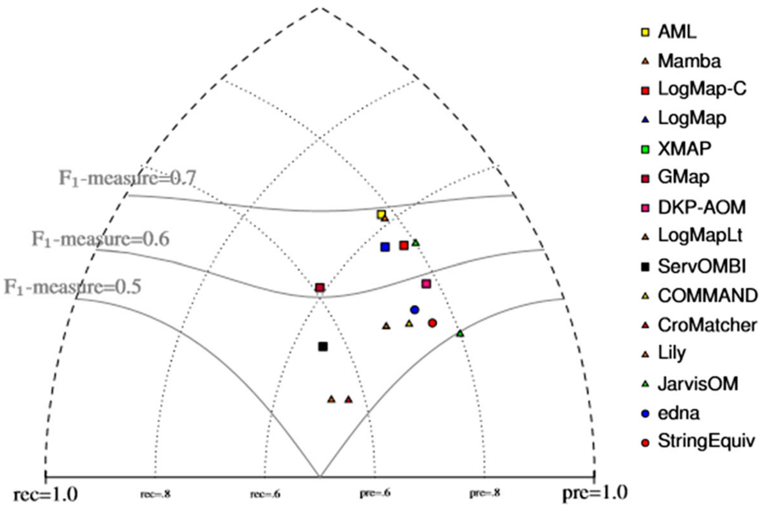


Fig. 13 DKP-AOM results on conference track ontologies (Cheatham et al. 2015)

LogMapLt) having better (or the same) results than both baselines in terms of highest average F1-measure. Group 2 consists of matchers (ServOMBI and COMMAND) performing better than baseline StringEquiv. Other matchers (CroMatcher, Lily, JarvisOM and RSDLWB) performed slightly worse than both baselines. We achieved F-Measure values better than the two Baselines results (edna, StringEquiv). Figure 13 presents the results obtained by running DKP-AOM on the Conference track of OAEI campaign 2015. Our system DKP-AOM has produced very competitive results among top ranked systems. Our precision measure is significantly high, recall is good, giving comparable F-measure value to depict a real effort towards detecting heterogeneities for the goal of ontology matching.

8.2 Evaluation based on uncertain version of reference alignment

The confidence values of all correspondences in the sharp reference alignments for the conference track are all 1.0. For the uncertain version of this track, the confidence value of a correspondence has been set equal to the percentage of a group of people who agreed with the correspondence in question (this uncertain version is based on reference alignment labelled as ra1). One key thing to note is that the group was only asked to validate correspondences that were already present in the existing reference alignments—so some correspondences had their confidence value reduced from 1.0 to a number near 0, but no new correspondence was added. Table 9 represents the evaluation based on the uncertain version of the reference alignments.

8.3 Evaluation based on violations of consistency and conservativity principles

DKP-AOM has given excellent performance for the evaluation based on the logical reasoning where oaei competition applied detection of conservativity and consistency principles violation. The consistency principle proposes that correspondences should not lead to unsatisfiable classes in the merged ontology, conservativity principle proposes that correspondences should not introduce new semantic relationships between concepts from one of input ontolo-

Table 9 Evaluation based on sharp, discrete uncertain and continuous uncertain metrics (Cheatham et al. 2015)

Matcher	Prec.	$F_{0.5}$-m.	F_1-m.	F_2-m.	Rec.	Inc. Align.	Conser. V.	Consist. V.
AML	0.78	0.74	0.69	0.65	0.62	0	39	0
Mamba	0.78	0.74	0.68	0.64	0.61	2	85	16
LogMap-C	0.78	0.72	0.65	0.58	0.55	0	5	0
LogMap	0.75	0.71	0.65	0.6	0.57	0	29	0
XMAP	0.8	0.73	0.64	0.58	0.54	0	19	0
GMAP	0.61	0.61	0.61	0.61	0.61	8	196	69
DKP-AOM	0.78	0.69	0.59	0.51	0.47	0	16	0
LogMapLt	0.68	0.62	0.56	0.5	0.47	3	97	18
edna	0.74	0.66	0.56	0.49	0.45	–	–	–
ServOMBI	0.56	0.56	0.55	0.55	0.55	11	1325	235
COMMAND	0.72	0.64	0.55	0.48	0.44	14	505	235
StringEquiv	0.76	0.65	0.53	0.45	0.41	–	–	–
CroMatcher	0.57	0.55	0.52	0.49	0.47	6	69	78
Lily	0.54	0.53	0.52	0.51	0.5	9	140	124
JarvisOM	0.8	0.64	0.5	0.4	0.36	2	27	7
RSDLWB	0.23	0.26	0.31	0.38	0.46	11	48	269

gies (Solimando et al. 2014). Our tool DKP-AOM is among five best tools which have no consistency principle violation (see Table 10), as we have employed various algorithms for the validation of initial mappings. The lowest number of conservativity principle violations has LogMap-C which has a repair technique for them. DKP-AOM has produce second-lowest number of conservativity principle violations, and employed algorithms to maintain conciseness and avoid redundancies in the resultant ontology (which is not in the scope of this paper). Conservativity principle violations can be favoured by redundancies, but those are not the only source of violations, due to possible complex interactions with other axioms in both ontologies. Further four tools have average of conservativity principle around 1.

Various versions of my system can be found at my personal site: http://sites.google.com/site/mhdfahad under plugins tab. The mapping system is separated from the merging system, and can be downloaded according to needs. For the merging of ontologies, use the same command of seals platform with –o following three paths, two for source ontologies and one for the output merged ontology. As a result of this command, a list of ontology mappings and a resultant merged ontology are produced.

9 Conclusion

Automatic mapping and merging of ontologies is vital to promote interoperability among multi-vender heterogeneous systems, where these different heterogeneous systems or their parts may possess heterogeneous but semantically overlapping knowledge representations in the form of ontologies. Mapping and Merging are crucial tasks also in ontology evolution and knowledge sharing scenarios, when it is rational to reuse existing knowledge, e.g. by modifying it according to varying requirements in order to better represent the modelling

Table 10 Statistics of consistency and conservativity principle violations

Matcher	#unsat.onto	#align	#incoh.align	#totalConser. Viol.	#avgConser. Viol.	#totConsist Viol.	#avgConsist. Viol.
LogMap-C	0	21	0	5	0.24	0	0
DKP-AOM	0	21	0	16	0.76	0	0
XMAP	0	21	0	19	0.9	0	0
JarvisOM	0	21	2	27	1.29	7	0.33
LogMap	0	21	0	29	1.38	0	0
AML	0	21	0	39	1.86	0	0

domain. This topic draws substantial attention within the research community, though it is not fully researched so far and new complex and effective solutions are needed. Therefore, this paper contributes the algorithms for the mapping and merging of class expressions of concepts for their merging in an automatic ontology merging system DKP-AOM. It addresses how to combine multiple axiomatic definitions into *one compact definition* considering the consistency of merged solution. The challenge of mapping axiomatic definitions is the most difficult task of merging concept definitions of the source ontologies. But, our presented algorithms for determining such mappings are reasonable and sound enough in order to enhance the effectiveness of the merging process. In addition, our initial results revealed higher precision and recall values of the whole process due to axiomatic mapping of class expression of concepts as it aims at identifying all the possible real mappings. We consider the quality criteria to avoid incoherence and inconsistency in the pre-integration phase of ontology merging so that the merged ontology should be free from semantic errors. In addition, by embedding this feature in our ontology merging system, one can get an ontology with the better quality as the combined rich axioms are added in the merged ontology. Our merging algorithm imports the first ontology as the merged ontology and then performs several operations (as described in our algorithms) to build the combined definitions of each of the concepts from the source ontologies. Each of the axiomatic definitions from the source ontologies are matched together, merging is performed on them, and the *combined rich axioms* are added in the merged ontology. Our merging algorithm performs deletion of axioms or the rewriting of some of them in order to preserve desired consequences while removing the undesired ones. Merging of axiomatic definitions really achieves a richer merged ontology which captures sufficient definitions from the source ontologies. We have also presented the results obtained by our DKP-AOM system within the OAEI 2015 campaign. This is our first successful participation in the Conference, OA4QA and Anatomy track of OAEI. DKP-AOM is participating with two versions (DKP-AOM and DKP-AOM_lite), DKP-AOM performs coherence analysis and has no consistency principle violation. In OA4QA track, DKPAOM out-performed in the evaluation and generated accurate alignments allowed to *answer all the queries* of the evaluation. Also, we can see its competitive results for the conference track in the evaluation initiative among other reputed systems. In the anatomy track, it has produced alignments within an allocated time and appeared in the list of systems which produce coherent results.

Acknowledgements I personally acknowledge Jimenez-Ruiz Ernesto for his supervision which gives me a lot of nice ideas to improve this article, and his efforts to give me help how to use alignment evaluation system (http://oaei.ontologymatching.org/) to run and test my system.

References

Alasoud A, Haarslev V, Shiri N (2009) An empirical comparison of ontology matching techniques. J Inf Sci 35(4):379–397

Arch-Int N, Sophatsathit P (2003) A semantic information gathering approach for heterogeneous information sources on WWW. J Inf Sci 29:357–374

Ba M, Diallo G (2012) ServOMap and ServOMap-lt results for OAEI (2012). In: Ontology matching workshop held with international semantic web conference 2012, vol 946. Boston, MA, USA

Ba M, Diallo G (2013) Large-scale biomedical ontology matching with ServOMap. IRBM 34(1):56–59

Bechhofer S, Harmelen FV, Hendler J, Horrocks I, McGuinness DL, Patel-Schneider PF, Stein LA (2004) OWL web ontology language reference. W3C Recommendation, 10 February (2004). http://www.w3.org/TR/owl-ref/

Bock C, Fokoue A, Haase P, Hoekstra R, Horrocks I, Ruttenberg A, Sattler U, Smith M, OWL 2 syntax document, OWL 2 Web ontology language structural specification and functional-style syntax. http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/

Bruijn JD, Ehrig M, Feier C, Martín-Recuerda F, Scharffe F, Weiten M (2006) Ontology mediation, merging and aligning. In: Davies J, Studer R, Warren P (eds) Semantic web technologies: trends and research in ontology-based systems. Wiley

Cheatham M, et al (2015) Results of the ontology alignment evaluation initiative 2015. In: The tenth international workshop on ontology matching, held with ISWC, Bethlehem (PA US)

Cheatham M, Hitzler P (2013) String similarity metrics for ontology alignment. Int Semant Web Conf 2:294–309

Cruz IF, Antonelli FP, Stroe C (2009) AgreementMaker: efficient matching for large real-world schemas and ontologies. PVLDB 2(2):1586–1589

Djeddi W, Khadir MT (2014) A novel approach using context-based measure for matching large scale ontologies. In: Proceedings of 16th international conference on data warehousing and knowledge discovery (DAWAK 2014), September 2–4. Springer, Munich, Germany, pp 320–331

Dou D, McDermott D, Qi P (2002) Ontology translation by ontology merging and automated reasoning. In: Proceedings of EKAW2002 workshop on ontologies for multi-agent systems', pp 3–18

Euzenat J, Shvaiko P (2007) Ontology matching. Springer, New York

Fahad M (2015) Engineering Semantic Web Ontologies by Ontology Merging and Classification. http://theses.univlyon2.fr/documents/lyon2/2013/muhammad_f#p=0&a=top

Fahad M, Moalla N, Bouras A (2012) Detection and resolution of semantic inconsistency and redundancy in an automatic ontology merging system. J Intell Inf Syst (JIIS). doi:10.1007/s10844-012-0202-y

Fahad M, Moalla N, Bouras A (2011) Towards ensuring satisfiability of merged ontology. In: Proceedings of 10th international conference on computation science, Singapore. ICCS 2011, Procedia Computer Science 4, pp 2216–2225

Guzman-Arenas A, Cuevas-Rasgado AD (2010) Knowledge accumulation through automatic merging of ontologies. Expert Syst Appl 37(3):1991–2005

Horridge M, Bechhofer S (2012) The OWL API: a Java API for OWL ontologies. Semant Web J 2(1):11–21

Huber J, Sztyler T, Nößner J, Meilicke C (2011) CODI: combinatorial optimization for data integration—results for OAEI 2011. In: CEUR workshop proceedings of the 6th international workshop on ontology matching, Bonn, Germany, RWTH, Aachen, pp 134–141

Jerroudi Z, Ziegler J (2008) iMERGE: interactive ontology merging. In: Proceeding of 16th international conference on EKAW, Italy

Jimenez-Ruiz E, Cuenca Grau B, Sattler U, Schneider T, Berlanga R (2008) Safe and economic re-use of ontologies: a logic-based methodology and tool support, ESWC 2008. LNCS 5021:185–199

Jimenez-Ruiz E, Cuenca Grau B (2011) LogMap: logic-based and scalable ontology matching. In: International seminor on web conference (ISWC), pp 273–288

Jiménez-Ruiz E, Cuenca-Grau B (2011) LogMap: logic-based and scalable ontology matching. The semantic web-ISWC 2011. Lect Notes Comput Sci 7031:273–288

Jiménez-Ruiz E, Cuenca-Grau B, Horrocks I, (2012) LogMap and LogMapLt results for OAEI (2012). In: Ontology matching workshop held with international semantic web conference 2012, vol 946. Boston, MA, USA

Jimenez-Ruiz E, Cuenca-Grau B, Horrocks I, Berlanga R (2009) Ontology integration using mappings: towards getting the right logical consequences In: Proceedings of 6th European semantic web conference, Heraklion, LNCS 5367, pp 475–479

Kim J, Jang M, Ha YG, Sohn JC, Lee SJ, Mo A (2005) OWL ontology merging and alignment tool for the semantic web. LNCS 3533(2005):116–123

Kotis K, Vouros GA, Stergiou K (2009) Towards automatic merging of domain ontologies: the HCONE-merge approach. J Web Semant 4(1):60–79

Kotis K, Katasonov A, Leino J (2012) AUTOMSv2 results for OAEI 2012. In: Ontology matching workshop held with international semantic web conference 2012, vol 946. Boston, MA, USA

Maiz N, Fahad M, Boussaid O, Bentayeb F (2010) Automatic ontology merging by hierarchical clustering and inference mechanisms. In: Proceedings of 10th international conference on KE and KM, (IKNOW) Austria, pp 81–93

McGuinness DL, Fikes R, Rice J, Wilder S (2000) An environment for merging and testing large ontologies. In: Proceedings of 7th international conference on principles of knowledge representation and reasoning, Colorado, USA, pp 483–493

Miller G (1995) Wordnet: a lexical database for English. Commun ACM 38(11):39–41

Mitra P, Wiederhold G (2002) Resolving terminological heterogeneity in ontologies. In: Proceedings of workshop on ontologies and semantic interoperability, 15th ECAI, Lyon, France, pp 45–50

Morphadorner (2010) http://morphadorner.northwestern.edu

Ngo D, Bellahsene Z (2012b) YAM++ results for OAEI 2012. In: Ontology matching workshop held with international semantic web conference 2012, vol 946. Boston, MA, USA

Ngo D, Bellahsene Z (2012a) YAM++ : a multi-strategy based approach for ontology matching task, knowledge engineering and knowledge management. Lect Notes Comput Sci 7603:421–425

Noy NF, Musen MA (2003) The PROMPT suite: interactive tools for ontology merging and mapping. IJHCS 59(6):983–1024

Pottinger R, Bernstein P (2003) Merging models based on given correspondences. In: Proceedings of 29th international conference on VLDB'03, Berlin, pp 862–873

Raunich S, Rahm E (2011) ATOM: automatic target-driven ontology merging. In: Proceedings of ICDE, pp 1276–1279

Raunich S, Rahm E (2012) Towards a benchmark for ontology merging. In: Proceeding of 7th OTM workshop on enterprise integration, interoperability and networking (EI2N'12), Rome, Italy

Shvaiko P, Euzenat J (2012) Ontology matching: state of the art and future challenges. IEEE Trans Knowl Data Eng 25(1):158–176

SimMetrics (2011) http://sourceforge.net/projects/simmetrics/

Solimando A, Jiménez-Ruiz E, Guerrini G (2014) Detecting and correcting conservativity principle violations in ontology-to-ontology mappings. Int Semant Web Conf 2:1–16

Sowa JF (1996) Ontologies for knowledge sharing. In: Manuscript of the invited talk at Terminology and Knowledge Engineering Congress (TKE '96), Vienna

Stumme G, Mädche A (2001) FCA-merge: bottom-up merging of ontologies. In: Proceeding of 7th IJCAI. Seattle, USA, pp 225–230

Visser PRS, Jones DM, Bench-Capon TJM, Shave MJR (1997) An analysis of ontological mismatches: heterogeneity versus interoperability. In: AAAI 1997 Spring symposium on ontological engineering, USA

Weizhuo Li, Qilin Sun (2015) GMap: results for OAEI 2015. In: Proceedings of the 10th international workshop on ontology matching collocated with the 14th international semantic web conference (ISWC)

Xie J, Liu F, Guan SU (2011) Tree-structure based ontology integration. J Inf Sci 37(6):594–613