Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 2216–2225

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

Towards ensuring Satisfiability of Merged Ontology

Muhammad Fahad[a, b,*], Nejib Moalla[a], Abdelaziz Bouras[a]

[a]*LIESP Lab, Cerral Center, Univesity of Lyon2, Bron, 69676, France*
[b]*CDSC lab, Muhammad Ali Jinnah University, Islamabad, 44000, Pakistan*

Abstract

The last decade has seen researchers developing efficient algorithms for the mapping and merging of ontologies to meet the demands of interoperability between heterogeneous and distributed information systems. But, still state-of-the-art ontology mapping and merging systems is semi-automatic that reduces the burden of manual creation and maintenance of mappings, and need human intervention for their validation. The contribution presented in this paper makes human intervention one step more down by automatically identifying semantic inconsistencies in the early stages of ontology merging. Our methodology detects inconsistencies based on structural mismatches that occur due to conflicts among the set of *Generalized Concept Inclusions,* and *Disjoint Relations* due to the differences between disjoint partitions in the local heterogeneous ontologies. We present novel methodologies to detect and repair semantic inconsistencies from the list of initial mappings. This results in global merged ontology free from *'circulatory error in class/property hierarchy'*, '*common class/instance between disjoint classes error', 'redundancy of subclass/subproperty relations', 'redundancy of disjoint relations'* and other types of '*semantic inconsistency'* errors. In this way, our methodology saves time and cost of traversing local ontologies for the validation of mappings, improves performance by producing only consistent accurate mappings, and reduces the user dependability for ensuring the satisfiability and consistency of merged ontology. The experiments show that the newer approach with automatic inconsistency detection yields a significantly higher precision.

Keywords: Ontology Mapping and Merging; Semantic heterogeneity; Inconsistency detection; Validation of mappings; Similarity Measure;

1. Introduction

Recent years have witnessed great development and successful use of ontologies for the knowledge representation, data annotation and information exchange between people, organizations, autonomous agents, web services or groups in open environments such as Semantic Web. But, as they are being developed for multiple purposes, needs, and requirements, same ontologies share overlapping domain knowledge and can be used for the annotation of multiple data sources such as web pages, xml repositories, relational databases, etc. [1]. The use of ontologies, as a means for providing a shared/common understanding of various domains, enables a certain degree of interoperation between these data sources. Thus, ontology alignment, mapping and merging systems have appeared to fulfill these demands, which are discussed in detail by [2]. They act as pillars in wide range of application domains and building collaborations that involve the sharing of data, knowledge and resources among

* Corresponding author. Tel.: +33 (0)4 78 77 26 27; fax: +33 (0)4 72 43 85 18.
E-mail addresses: firstname.lastname@univ-lyon2.fr.

 Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.242

modern companies, and aid in developing a new ontology by reusing existing open ontologies.

The problem of ontology matching and merging is very hard to perform manually beyond a certain degree of complexity, size, and number of ontologies, and need fully automatic methodologies for enabling interoperability in the dynamic environment such as the semantic web and data warehouse where analysis context is made on-the-fly. Although, there is a great effort seen, but, still state-of-the-art ontology mapping and merging systems is semi-automatic that reduces the burden of manual creation and maintenance of mappings and needs human intervention for their validation. These systems and approaches use different aids such as common vocabulary, reference ontology, basic initial alignments by expert, etc., each of which might be appropriate in some tasks with given set of circumstances, but are not feasible for the dynamic environments [2]. Recent studies on ontology merging show that due to conceptualization and explication mismatches between local ontologies, fully automatic merging is unattainable [3]. But, effective algorithms for computing semantic correspondences help us reach at a position, where ontology merging can be carried out with minimum human intervention. In addition, in a study about the gap analysis between ontology mapping and merging techniques, [4] reported, "*The gap identified here suggests that research is required to find ways through which different conceptualization mismatches can be detected and resolved in order to give accuracy to the process of mapping and thus verifying the knowledge being shared*".

The contribution presented in this paper minimizes human involvement one step more down during the ontology merging process and presents a novel methodology for the detection of semantic inconsistencies in the initial stages of ontology merging. Disjoint knowledge analysis and preservation in ontology merging helps to identify the conceptualization mismatches between heterogeneous ontologies and provides more accuracy to the process of mapping. Our ultimate goal is to check the semantic correctness and consistency of mappings, and ensure the satisfiability of merged ontology (as this will act as a fundamental analysis context later on in the semantic application). In order to achieve this, our methodology detects semantic inconsistencies from the list of initial mappings by exploiting *Generalized Concept Inclusions* (GCIs), and *Disjoint Knowledge Axioms* (DKA) present in local ontologies. It checks whether lexically same concepts within the local source ontologies must not contradict each other with respect to set of *GCIs* or *DKA*. The consistency checker module acts as a filter at the initial merging stage checking for a set of basic conditions before allowing axioms to be added to the global ontology. Due to the use of more semantics present in the source ontologies and test criteria for the validation of mappings, our approach enhances the accuracy of mappings, and produces consistent, complete and coherent global ontology.

The rest of paper is organized as follows. Section 2 discusses state-of-the-art. Section 3 discusses an overview of our automatic ontology merger system, *DKP-AOM*. Section 4 presents semantic inconsistencies due to the conflicts among *GCIs* and *DKA* in local heterogeneous ontologies, and discusses algorithms for their detection. Section 5 shows our experiment results. Finally, section 6 concludes the paper and shows future directions.

2. Related Works

In the research literature, there are many diverse approaches, techniques and systems for alignment, mapping and merging of heterogeneous ontologies. An instance based methodology, FCA-Merge, employs formal concept analysis to make the concept lattice, and considers concepts having the identical instance candidate for merge [5]. IF-Map, also, uses the formal analysis of concepts and gets aid from the common reference ontology that comprises common vocabulary about the local subject ontologies for their mapping [6]. GLUE follows a hybrid strategy making use of instance and taxonomic structure matching techniques with machine learning to determine the probabilities of concepts for the ontology integration [7]. OLA exploits distance based algorithms and alignment API for finding the correspondences between OWL-Lite ontologies by making use of all the elementary matching techniques [8]. QOM aimed at gaining efficiency by dynamic programming approach rather then effectiveness by matching algorithms. It avoids the complete pair-wise comparison of concepts and employs many heuristics for choosing only the best candidate mappings, and thus reduces the runtime complexity of matching operation [9].

The interactive ontology merging tools, PROMPT suite [10] and Chimaera [11], exploit syntactic concept label matching techniques and to some extent the structure of local ontologies for the initial comparisons, and merge ontologies on user feedback. ONION uses the structure of taxonomy, local definitions and formalization of articulation ontologies for the merging of different ontologies [12]. IMerge exploits visual analytics techniques for helping users in the process of ontology merging. It employs various interactive visualizations facilitating end-user about why and where concepts can be merged [13]. HCONE-merge makes use of the latent semantic indexing

mechanisms for computing possible mappings and then requires human intervention for their validation. Then, it employs reasoning services of DL for the automatic merging of local ontologies [3].

Besides above, very few recent works exploit semantic based techniques. CtxMatch [14], S-Match [15], aim at determining semantic matching and inconsistency detection between the concepts of ontologies. They transform concept descriptions into *Description Logic* (DL) axioms, i.e., change matching problem into a propositional unsatisfiability problem. Then, they make use of available open source DL reasoners for finding the semantic relations (e.g., equivalence, subsumption) between the concepts that correspond semantically. Niepert et al. provide a novel probabilistic-logical framework for ontology matching that employs a wide range of matching strategies. Their ontology matching framework is based on Markov logic which combines first-order logic and undirected probabilistic graphical models [16]. For inconsistency detection in the process of ontology alignment, [17], [18] and [19] propose correcting inconsistent alignments. Meilicke et al. aim at finding consistent alignments and feeding them back to the matcher to find new alignments [17]. Ruiz et al. exploit a revision operator for achieving the consistent result by modifying an alignment between the two ontologies [18]. In the field of argumentation, Trojahn et al. provide an approach for deciding which correspondences to preserve or reject in the ontology matching [19]. ASMOV matching algorithm exploits the lexical and structural analysis for determining the correspondences between ontologies. Then, it verifies the determined correspondences by the use of formal semantics to compute whether they comply with the desired characteristics [20].

Our analysis on research literature is that the approaches like [14], [15], [16], [17], [18], [19], [20], consider consistency issues or semantic-based technique for ontology matching process but did not extended their works for the automatic ontology merging. Interactive tools such as [3], [11], [12], [13] for the ontology merging are not semantic-based, don't consider consistency issues and require human intervention for the validation of mappings. Our initial effort towards ontology merging was also a semi-automatic system, Disjoint Knowledge Preservation in Ontology Merging (*DKP-OM*). *DKP-OM* follows a hybrid approach for the detection of mappings and employs test criteria to filter out inconsistent mapping in the process of ontology merging [21], [22]. The consistent correspondences act as a list of suggestions to the end user and then system requires human intervention for the generation of global merged ontology. Another approach for automatic ontology merging is formalized based on the hierarchical clustering and inference mechanisms in which semantic inconsistencies are neglected during the generation of merged ontology [23]. In this paper, we are working for overcoming both the limitations of existing works, i.e., reducing the human effort for providing the feedback and automation of inconsistency detection inside the ontology merging system. This paper presents *DKP-AOM* which extends the methodology of semi-automatic *DKP-OM* system to encounter more structural and semantic conflicts, and contribute more optimized fully automatic solution for accessing and resolving semantic consistencies in an ontology merging. *DKP-OM* detects the semantic inconsistencies with the help of hidden global merged ontology that is formed by leading initial mappings. Then, it applies consistency criteria on the hidden merged ontology to ensure its satisfiability. Making hidden ontology, consumes time and resources, and adds performance overheads to the overall performance of the system. The newer developed *DKP-AOM* system employs various algorithms to detect inconsistent mappings from the initial list of mappings, and saves time and resources for the generation of hidden intermediate global ontology.

3. Overview of DKP-AOM: An Automatic Ontology Merging System

DKP-AOM system is composed of three main components (see Fig 1). First, it generates the intermediate models (OWL-DL Graphs) of source ontologies using *Jena* API. Using these graphs, *MatchManager*, which comprises a set of individual matching algorithms, performs the first level task of finding the initial linguistic, synonym and axiomatic based mappings between concepts. Linguistic analysis of concept labels and properties is done with the MorphAdorner[a] (version 1.0). Synonym similarity is detected with the help of JWNL[b] (version 1.4.1) that aids accessing the WordNet[c] (version 3.0) dictionary. MorphAdorner is helpful in various cases especially the lemmatization process is worth useful for detecting the base words of terms and irregular verbs used in local

[a] http://morphadorner.northwestern.edu

[b] http://sourceforge.net/projects/jwordnet/

[c] http://wordnet.princeton.edu

ontologies. For example, concept "*students*" to lemma "*student*" and properties ("*Accepted*", "*accepting*", "*Accept*", and "*accepts*") to their base "*accept*". *MatchManager* propagates the initial mappings to *ConsistencyChecker* for their validation. *ConsistencyChecker* has many detectors that make the validation of each mapping found in the initial stage so that the merged ontology stays consistent with reference to the source ontologies. When the initial mappings pass the consistency test, *ConsistencyChecker* passes the mappings to the *Reasoner*. Finally, *Reasoner* aggregates the output of different similarity measures, resolves conflicts and merges initial mappings to generate global merged ontology. Finally, it compiles the output as merged global ontology automatically or final list of consistent mappings as required by the end user. In this step, it ensures the ultimate goal of achieving the satisfiability of merged ontology by checking the correctness and consistency of concepts, properties, and axioms of the generated global ontology. For semi-automatic generation of global ontology from mapping list, it shows the semantically consistent mappings to the user as a list of initial suggestions, and asks for the user feedback. In this case, it follows the cyclic approach as other merging systems (e.g., Prompt) to generate a global merged ontology.

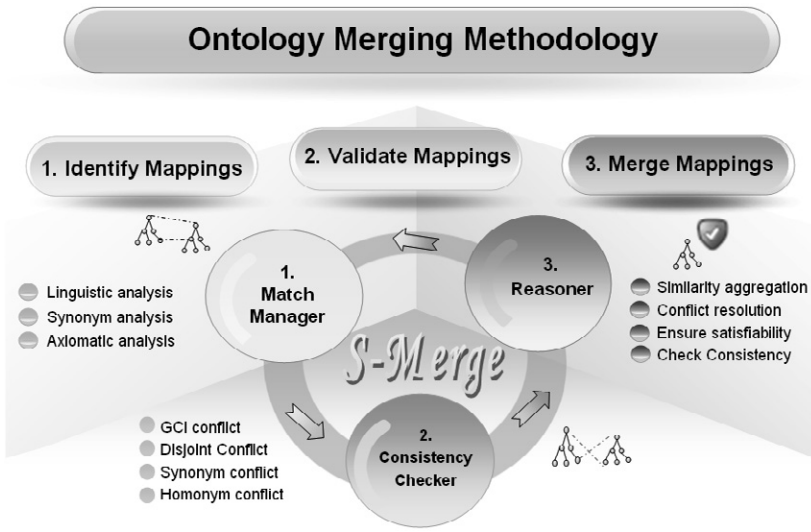

Fig. 1. Semantic Ontology Merging Methodology of DKP-AOM

4. Ensuring Satisfiability of Global Merged Ontology

The main goal of *ConsistencyChecker* module of *DKP-AOM* is to ensure the satisfiability of global merged ontology that is generated by following initial mappings, so that *Tbox* (Terminological box) and *Abox* (Assertional box) of global ontology comprise of consistent set of *Generalized Concept Inclusions* (e.g., GCI: C $\sqsubseteq$ D stays consistent according to local ontologies). Formally, the satisfiability of global merged ontology is expressed as:

Definition 1: A *Global Ontology GO* is a pair $GO = (\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is a *Tbox* and $\mathcal{A}$ is an *Abox*, and *Tbox* $\mathcal{T}$ is generated from the *Mapping_list* {Mapping(C_1,D_1), Mapping(C_2,D_2),…,Mapping(C_n,D_n)} where C_1,C_2, ...,C_n belong to local ontology O_1 and D_1,D_2, ...,D_n belong to local ontology O_2. An interpretation $\mathcal{I}$ is a model for *GO* if it is both a model for $\mathcal{A}$ and $\mathcal{T}$. A Global Ontology *GO* logically implies α, where alpha is either an Abox statement (i.e., concept or role instantiation) or a *Tbox* statement (i.e., concept introduction), written $GO \models \alpha$, iff α is satisfied by every model of *GO*.

Several inconsistency detectors, inside the *ConsistencyChecker* component, are responsible for finding the semantic inconsistencies in the initial mappings found. Each of the detectors works independently employing specific algorithm so that only consistent and accurate mappings are generated to build satisfiable merged ontology. When a detector discovers any inconsistent mapping, it notifies to the *ConsistencyChecker* that warns about the inconsistent situations, which could occur in merged global ontology by following inconsistent initial mapping. Hence, it reduces the human intervention by validating the merged ontology automatically. There are various types of inconsistencies, incompleteness and redundancies that may occur in an ontology (as discussed in [24]), and in merged ontology as well. We aim at building a quality criteria based on consistency, completeness and conciseness

for the merged ontology, but in this paper, we are only addressing the semantic inconsistencies and redundancies that occurs due to the conflicts between *GCIs* and *DKA*.

4.1. Semantic Inconsistency due to GCIs Conflict

Global Merged Ontology, GO, is meant to be semantically consistent when its *Tbox* satisfies all the set of *Generalized Concept Inclusions* (*GCIs*) of local heterogeneous ontologies, and all the concepts follow the subsumption criteria with respect to knowledge in local ontologies. When *GCIs* of local heterogeneous ontologies conflict with each other, then the global merged ontology suffers from various types of semantic inconsistencies. Formally, the following definitions express the subsumption rule for global ontology and inconsistent mappings.

Definition 2: For *Global Ontology GO*, an interpretation $\mathcal{I}$ satisfies a GCI C ⊑ D, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and an interpretation is a model of a *Tbox* $\mathcal{T}$, if it satisfies all *GCIs* in the *TBox* $\mathcal{T}$.

Definition 3: Mapping(C,C'), Mapping(D,D') are semantically inconsistent as they violate the subsumption rule with respect to *GCIs* of local ontologies O_1 and O_2, such that in O_1, C ⊑ D w.r.t. local *Tbox* $\mathcal{T}_1$ & in all the models of $\mathcal{T}_1$ $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, And in O_2, D'⊑ C' w.r.t local *Tbox* $\mathcal{T}_2$ & in all the models of $\mathcal{T}_2$ $D'^{\mathcal{I}} \subseteq C'^{\mathcal{I}}$.

There are several possibilities which create semantic inconsistency in global merged ontology. For example, let O_1 and O_2 be two source ontologies (see Fig 2) of software engineering domain comprising workers following the Object Oriented (OO) approach or Structured Engineering approach for developing programs for the companies. Semantic inconsistent mappings (or *GCIs* conflict) mean a situation where a parent concept in O_1 maps on a child concept of O_2 and a child concept of O_1 maps on a parent concept of O_2. In this case, initial mappings by *MatchManager* subcomponent, i.e., $\text{Mapping}_{O1,O2}$(*Designer*, *Designer*) and $\text{Mapping}_{O1,O2}$(*OOknow_Associate*, *OOKnow_Associate*), create *GCIs* conflict.

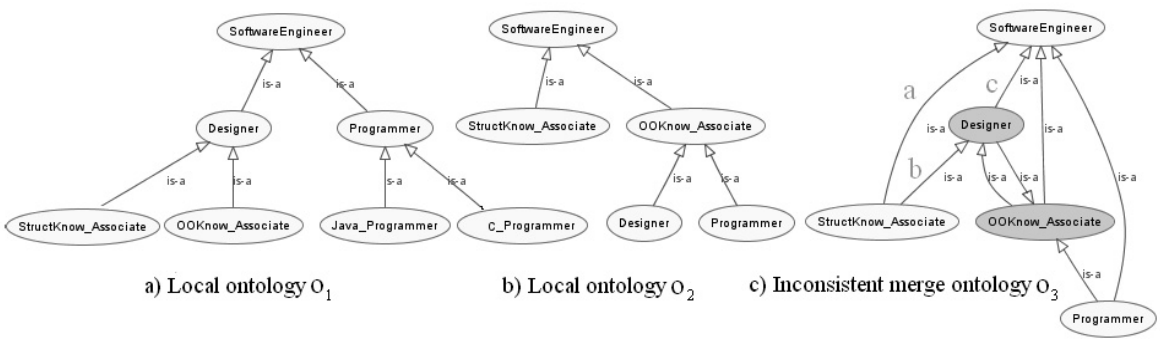

Fig. 2. Local ontologies O_1 and O_2 lead to Inconsistent Global Merged Ontology O_3

Our system works on these source ontologies as follows. It gets the source ontologies, makes the intermediate models for similarity computation, and annotates depth with each concept in the ontology hierarchy. *MatchManager* computes correspondences between the ontology concepts and produces initial mappings between source ontologies as follows: $\text{Mappings}_{O1,O2}$\{(*SoftwareEngineer*$_1$, *SoftwareEngineer*$_1$), (*Programmer*$_2$, *Programmer*$_3$), (*Designer*$_2$, *Designer*$_3$), (*StructKnow_Associate*$_3$, *StructKnow_Associate*$_2$), (*OOKnowledge_Associate*$_3$, *OOKnowledge _Associate*$_2$)\}. Each of the mappings contains names of classes from O_1 and O_2 and depths of concepts in the ontology associated with it (here shown as subscript in $\text{Mappings}_{O1,O2}$). It propagates these mappings to *ConsistencyChecker*, which ensures correctness and consistency issues. *ConsistencyChecker* applies the test criterion (shown in the Fig 3) to ensure the semantic correctness of mappings. It detects the *GCIs* conflict that lead to inconsistency from the depth associated in the mapping list and not by traversing the source ontologies to reduce the time complexity of validating initial mappings. Otherwise, traversing the ontologies again for detecting inconsistencies would be very much costly and compromises the performance of the system. First, it extracts mappings with unequal depth from the list of initial mapping as they are ambiguous due to the depth differences, but these may or may not create inconsistencies in merged ontology. Then, it validates whether selected mappings lead to inconsistencies, if they have parent-child relationship between them. It is a necessary condition for semantic

inconsistency caused by *GCIs* conflict that the mappings with unequal depth have parent-child relationship, which lead to cycles in the merged ontology. In this example, *ConsistencyChecker* detects $Mapping_{O1,O2}$(*Designer*$_2$, *Designer*$_3$) and $Mapping_{O1,O2}$(*OOKnowledge_Associate*$_3$, *OOKnowledge_Associate*$_2$) as inconsistent mappings that create the semantic inconsistency. As *Designer* in O_1 ontology lexically corresponds to *Designer* in O_2 ontology, and *OOKnowledge_Associate* in O_1 corresponds to *OOKnowledge_Associate* in O_2, and these mappings fulfil the condition of parent-child relationship. But, the mapping pairs (3 and 4), (2 and 4), and (2 and 5) in $Mappings_{O1,O2}$ have unequal depth but they do not fulfil the condition of parent-child relationship between them, and hence they cannot create inconsistencies in the merged ontology.

```
Program: Integer Test_GCIConflict( mappingList, O1, O2)
1   START
2      FOREACH mp in mappinglist
3            IF( mp.depth1 != mp.depth2) THEN
4                  selectedMappingList.add(mp);
5      END FOREACH
6      n = selectedMappingList.length;
7      FOR ( i=0; i<n-1; i++) DO
8           Mapping mp1 = selectedMappingList( i );
9           FOR (j=i+1; j<n; j++) DO
10             Mapping mp2 = selectedMappingList( j );
11             IF (mp1.depth1<mp2.depth1 and mp1.depth2>mp2.depth2) THEN
12                    IF(O1.CheckParentChildRelation(mp1.name1, mp2.name1)AND
13                           O2.CheckParentChildRelation(mp2.name2, mp1.name2))
14                                  THEN
15                                         Return 1; //semantic consistency found
16                           ELSE Return 0;
17          End Inner For
18     End outer For
19   END PROGRAM
```

Fig. 3. Algorithm for the detection of GCI conflict from the list of initial mappings

This type of inconsistency present in initial mappings leads to *redundancies* and various other types of *inconsistencies* (i.e., *'circulatory error in class hierarchy error', 'wrong subsumption inconsistencies'*) in merged global ontology as indicated by the cases 2 and 3 in Table 1. *Redundancy of subclass-of error* can be observed in O_3 by analyzing *subclass-of* axiom between (*Structknow_Associate*, *SoftwareEngineer*), already having another *subclass-of* axioms between (*Structknow_Associate*, *Designer*) and (*Designer*, *SoftwareEngineer*) as indicated by *IS-A* relationships (a, and b, c) in the Fig 2. Therefore, it is very crucial to identify all these scenarios of semantic inconsistencies and repair them before building the global merged ontology.

Table 1. Various cases of Semantic Inconsistency due to GCIs conflict among ontologies

S. No.	Let $\{A, B, C, \ldots, L\} \in O_1$ and $\{N, M, O, \ldots, Z\} \in O_2$
Case 1.	Mapping (A,M) and Mapping (B,O) create semantic inconsistency when in O_1 $A \sqsubseteq B$, and in O_2 $O \sqsubseteq M$
Case 2.	Mapping (A,M), Mapping (B,N) then Mapping (C,O) create redundant subsumption when in O_1 $C \sqsubseteq B \sqsubseteq A$, and in O_2 $N \sqsubseteq M, O \sqsubseteq M$, but $O \sqsubseteq \neg N$ (here C,B,O,N are not satisfiable in GO) it is a special kind of satisfiability conflict, This type of satisfiability conflict is important because it creates redundant subsumptions.
Case 3.	Mapping (A,M) and Mapping (B,O) create cyclic inconsistency when in O_1 $A \sqsubseteq B$, and in O_2 $O \sqsubseteq M$

4.2. Semantic Inconsistency due to Alignment Conflict among Disjoint Relations

Global Merged Ontology GO is free from alignment conflict among disjoint relations when all the concepts in its TBox satisfy all the set of axioms present in local heterogeneous ontologies, and all the concepts follow the satisfiability criteria with respect to disjoint or overlapping knowledge in local ontologies. Formally, the following definitions express the satisfiability of concepts in global ontology and semantically inconsistent mappings.

Definition 4: A concept C in global merged ontology *GO* that is generated by Mapping (C, C') from local ontologies O_1, O_2 is satisfiable w.r.t a *TBox* $\mathcal{T}$ if there exist a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$, and *TBox* of global

merged ontology is satisfiable if it admits a model.

Definition 5: *Mapping(C,C')*, *Mapping(D,D')* suffers from alignment conflict among disjoint relations as they violate the satisfiability rule with respect to knowledge in local ontologies O_1 and O_2, Given that in O_1, C *disjointWith* D w.r.t a local Tbox $\mathcal{T}_1$ & in all models of $\mathcal{T}_1$ $C^{\mathcal{I}}$ *disjointWith* $D^{\mathcal{I}}$, But in O_2, C'*overlapping* D' w.r.t a local Tbox $\mathcal{T}_2$ & in all models of $\mathcal{T}_2$ $C^{\mathcal{I}}$' *overlapping* $D^{\mathcal{I}}$'.

For an example, consider local ontologies in Fig 4, where *Designer* and *Programmer* are disjoint concept in software engineer ontology O_1 to avoid the situation where same person may design and program in a wrong way, but in ontology O_2 they are overlapping concepts and have a common class Tester between them. In this scenario, *MatchManager* suggests the following three Mappings; $Mappings_{O1,O2}\{(SoftwareEngineer_1, SoftwareEngineer_1), (Programmer_2,Programmer_2), (Designer_2, Designer_2)\}$. Then, *ConsistencyChecker* applies the test criteria for the consistency analysis of initial mappings and employs the algorithm for the detection of alignment conflict among disjoint relations. In this example, the initial $Mapping_{O1,O2}$*(Designer, Designer)* and $Mapping_{O1,O2}$ *(Programmer, Programmer)* create alignment conflict among disjoint relations as *Designer* and *Programmer* are disjoint in O_1 but there is a common class between them in O_2.

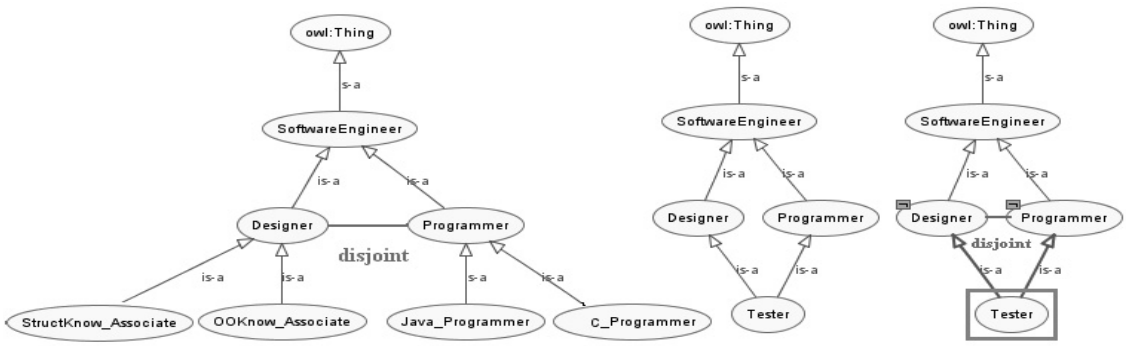

Fig. 4. Ontologies O_1 and O_2 lead to an inconsistent merged ontology O_3

There are several possibilities (see table 2) of alignment conflict among disjoint relations between local ontologies that occur when two concepts in source ontology O_1 are disjoint but overlapping in ontology O_2, i.e., there is a common class (case 1), equivalence relation (case 2), parent-child relationship (case 3), common instance (case 4) between them. There can be a situation (case 5) where concepts are not specified as disjoint in O_2, i.e., it may be a disjoint knowledge omission between them or cannot be considered as disjoint due to the open world assumption. The *Test_DisjointConflict* (see Fig 5) analyzes the mappings belonging to the concepts having disjoint axioms among them. When it gets the mapping (e.g., 2) of concept (e.g. O_1*:Programmer*) having disjoint axiom, then it analyzes it's disjoint concepts (e.g., *Designer* in this case), and find their mappings (e.g., O_1*:Designer*, O_2*:Designer*) in the mapping list $Mappings_{O1,O2}$. Then, it checks whether the disjoint concepts in O_1 are overlapping in ontology O_2. In case of inconsistency, system warns the situation and does not proceed merging with these mappings as they lead to '*common class between disjoint classes error*'. Then, it follows the repair mechanism by placing the common class or preserving the disjoint knowledge in the merged ontology.

Table 2. Various cases of Semantic Inconsistency due to DKA conflict among ontologies

S. No.	Mapping Rules when $\{A, B, C, \ldots, L\} \in O_1$ and $\{N, M, O, \ldots, Z\} \in O_2$
Case 1.	Mapping (A,M) and Mapping (B,O) are inconsistent when in O_1 $A \sqsubseteq \neg B$, and $\exists$ X (concept) in O_2 such that $X \sqsubseteq M$, $X \sqsubseteq O$
Case 2.	Mapping (A,M) and Mapping (B,O) are inconsistent when in O_1 $A \sqsubseteq \neg B$, and in O_2 $M \equiv O$
Case 3.	Mapping (A,M) and Mapping (B,O) are inconsistent when in O_1 $A \sqsubseteq \neg B$, and in O_2 $M \sqsubseteq O$
Case 4.	Mapping (A,M) and Mapping (B,O) are inconsistent when in O_1 $A \sqsubseteq \neg B$, and $\exists$ {x} (instance) in O_2 such that $\{x\} \in M$ & $\{x\} \in O$
Case 5.	Mapping (A,M) and Mapping (B,O) are inconsistent when in O_1 $A \sqsubseteq \neg B$, and in O_2 M notDisjointWith O

```
Program: Integer Test_DisjointConflict( mappingList, O1,O2)
1 START
2   n = mappingList.length;
3   FOR( i=0; i<n; i++) DO
4     Mapping mp = mappingList(i);
5       IF (HasDisjointConcepts(mp.name1) ) THEN
6          Discon[] = GetDisjointSiblings(mp.name1);
7          FOR (j=0; j<Discon.length; j++) DO
8             Mapping mp2 = SearchMapping(Discon[j]);
9             IF ((mp.name2 & mp2.name2) are NotDisjointwith OR   Equivalence OR
10              ParenChildRelationship OR CommonClass OR CommonInstance )  THEN
11                    Return 1;
12                ELSE Return 0;
13            END IF
14         END For
15      END IF
16    END FOR
17 END PROGRAM
```

Fig. 5. Algorithm for the detection of alignment conflict among disjoint relations from the list of initial mappings

5. Discussion and Experiment Results

We have observed that these inconsistent situations are very much common dealing with the real world heterogeneous ontologies due to inherent semantic conflicts that came out from different communities over the internet. These inconsistencies between local ontologies can occur during the mappings of properties to build the property hierarchies in the merged ontology. Our approach employs the similar rules as described above for the detection of inconsistencies among property hierarchies. But, their occurrences in property hierarchies (datatype and object property hierarchies) are very less observed in real ontologies. However, in all these scenarios, system does not automatically follow the initial mappings to build global merged ontology. The system should detect the erroneous situations and employ repair mechanisms for building the merged ontology. The time complexity of presented test criterion for the detection of disjoint conflict is O(n*d) and GCI conflict is O(n), where n is the number of mappings in the mapping list and d is the number of disjoint axioms in an ontology. These tests give good performance working with the real world corpus. As compared to Prompt [9], Chimera [10], I-Merge [4], OMerSec [26], our ontology merger performs a step ahead for building automatic merged ontology employing repair mechanisms, also with informing the end user about the situation and/or inconsistencies during the building of automatic merged ontology. The concept of instant validation of initial mappings, for the users who are not much experts in building ontologies, but, interested to build ontologies for their domain by reusing the existing several domain ontologies is highly useful. Otherwise, the user may face various consequences as he may not be familiar with ontological errors of these kinds.

We have evaluated the quality of automatically generated merged ontology by performing different type of experiments on web ontologies that belong to 4 different categories, i.e., *University*, *Publication*, *Book*, and *Conference*. To ensure the significance of disjoint axiom analysis for ensuring satisfiability and consistency checking, we added various disjoint axioms in these ontologies manually at disjoint partitions. Finally, the OWL ontology constructs in global merged ontology are manually checked to measure the consistency, completeness and coherency. The experiment result can be positive or negative depending on the algorithm's accuracy. The experiment results for each concept may or may not match with the manual expert discussion, resulting four different cases. First, True Positive (TP) or Correct Mappings, which means that OWL Concept C_a is correctly mapped on C_b in merged ontology and Human Expert is agreed with it. Second, True Negatives (TN) or Correct Not-Mappings, which means that OWL Concept C_a is not mapped on C_b in merged ontology and Human Expert is agreed with it. Third, False Positives (FP) or Incorrect Mappings, which means that OWL Concept C_a is incorrectly mapped on C_b in merged ontology and Human Expert is not agreed with it. Fourth, False negative (FN) or Missed Mappings, which means that OWL Concept C_a is not mapped on C_b in merged ontology and according to Human Expert it should be mapped. On the basis of these four cases, *Precision (TP/TP+FP)* and *Recall (TP/TP+FN)* values are calculated. Fig 6 shows the comparative analysis between the hybrid approach with exploiting consistency

detection and simple approaches without employing test criterion. The result of our hybrid approach is promising as the test criterion has rejected the initial incorrect linguistic mappings. The experiments show that the newer approach with automatic inconsistency detection yields a significantly higher precision. In addition, use of *MorphAdorner* that reduces inflected spellings of concept labels to its lexical root or lemma form is highly useful to cope with the semantic heterogeneities in local ontologies. When the precision is equal to 1, it means that there are no False Positives, i.e., concepts that should be mapped between local ontologies are truly mapped by the approach. But, when the precision is low, it means that the approach has made some concept mappings which it should not made. Similarly, when the recall is equal to 1, it means that there are no False Negatives, i.e., concept mappings made by approach are actually be made. But, when the recall is low, it means that the approach has missed some concept mappings that it should made according to the human expert. By manual inspection, we conclude that the incorrect or missed mappings are due to the concepts labels formed from composite words or defined by technical terms, and their detection is beyond the power of linguistic or synonym strategies.

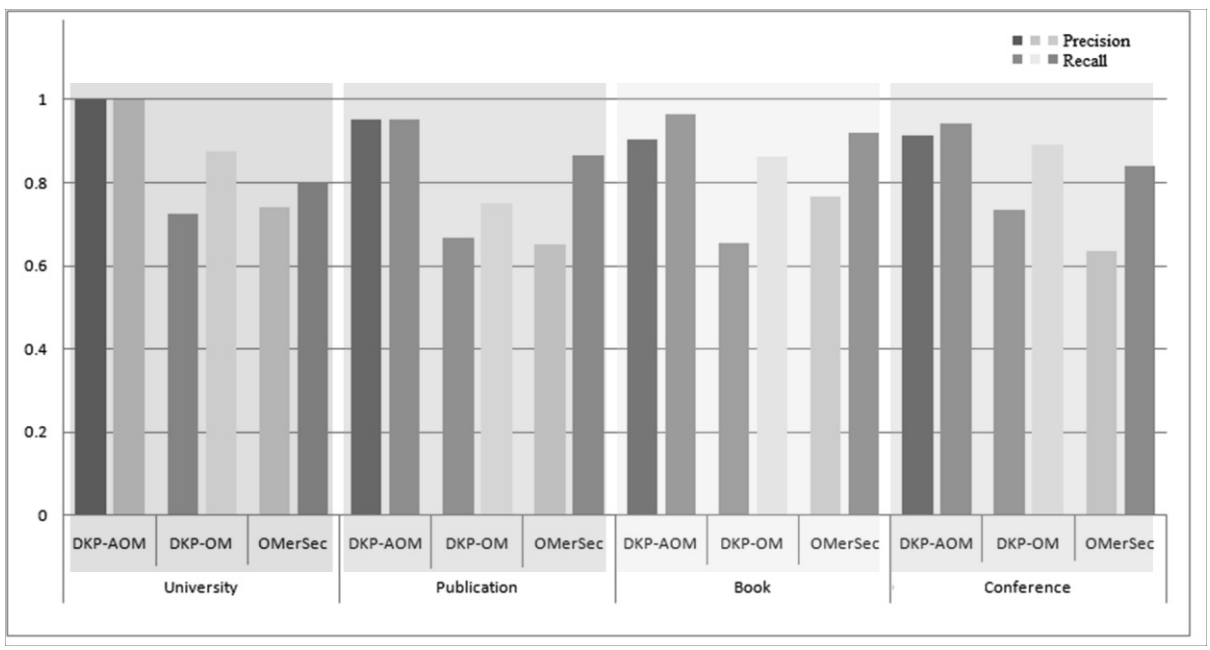

Fig. 6. Comparing Precision and Recall of DKP-AOM with other ontology merging systems

6. Conclusion and Future Directions

In this paper, we have presented our methodology of *DKP-AOM* system that exploits linguistic, synonym and axiomatic matching to find correspondences between concepts. In addition, it employs test criterion for the detection of semantic inconsistencies that originates when concepts in local ontologies contradict according to their subsumption or disjointness criterion. In this way, it detects explication and conceptualization mismatches between heterogeneous ontologies and promotes a larger pool of knowledge and information to be integrated to facilitate new reliable communication and reuse. The detection of conflict among *Generalized Concept Inclusions* between local ontologies would result global ontology free from '*circulatory error in class/property hierarchy*', '*redundancy of subclass/subproperty of relations*' errors and other types of 'semantic inconsistency' that occur due to wrong placement of concept in merged ontology from local heterogeneous ontologies. Similarity, detection of alignment conflict among disjoint relations between local ontologies would result global ontology free from '*common class/instance between disjoint classes*' and '*redundancy of disjoint relations*' errors. Our algorithm detect inconsistency from initial mappings in the early stages of ontology merging so that only consistent mappings would result global merged ontology with consistent set of axioms. Early automatic detection of inconsistency not only saves time and resources, but also lessons user intervention for ensuring the consistency of merged ontology. We

have implemented the algorithm of detecting such inconsistencies and evaluated the working of system on real world ontologies. The outcomes are very interesting by embedding inconsistency detection algorithms inside the ontology merging system in terms of precision of results, reduction of human expert dependability, computational efficiency, and good level of automatic consistency checking, etc. One of our ongoing researches is to apply a variety of architectural, optimization, and design principle to improve the performance of our system, and further enhance it based on other test criteria based on consistency, completeness and conciseness (as presented in [24]) that lead towards accurate merged ontology with avoidance of higher level of user intervention in the merge process.

References

1. M. Klein, Combining and relating ontologies: an analysis of problems and solution, In Proc. of Workshop on Ontologies and Information Sharing, Seattle, USA (2001) 53-62.
2. J. Euzenat and P. Shvaiko, Ontology Matching, Springer-Verlag, Berlin (2007).
3. K. Kotis, G.A. Vouros, K. Stergiou, Towards automatic merging of domain ontologies: The HCONE-merge approach, Web Semantics: Science, Services and Agents on the World Wide Web, 4(1) (2006) 60-79.
4. N. Anjum, J. Harding, B. Young, K. Case, Gap Analysis of Ontology Mapping Tools and Techniques, Enterprise interoperability, (2010) 303-312.
5. G. Stumme and A. Mädche, FCA-merge: bottom-up merging of ontologies, In Proc. 7th IJCAI, Seattle, USA, (2001) 225-230.
6. Y. Kalfoglou and M. Schorlemer, If-map: an ontology mapping method based on information flow theory, JODS 2800 (2003) 98-127.
7. A. Doan, J. Madhaven, P. Domingos, A. Halevy, Ontology matching: A machine learning approach, Handbook on Ontologies in Information Systems, Springer-Verlag (2004) 397-416
8. J. Euzenat. and P. Valtchev, Similarity-based ontology alignment in owl-lite, In Proc. 16th ECAI-04, Spain (2004) 333–337.
9. M. Ehrig, and S. Staab, QOM- Quick Ontology Mapping, In Proc. 3rd ISWC, LNCS 3298, Springer-Verlag, (2004) 683-696.
10. N.F. Noy and M.A. Musen, The PROMPT suite: interactive tools for ontology merging and mapping, IJHCS 59(6) (2003) 983-1024.
11. D.L. McGuinness, R. Fikes, J. Rice, S. Wilder, An environment for merging and testing large ontologies, In Proc. 7th International Conference on Principles of Knowledge Representation and Reasoning, Colorado, USA (2000) 483-493.
12. P. Mitra and G. Wiederhold, Resolving Terminological Heterogeneity in Ontologies, In Proc. Workshop on Ontologies and Semantic Interoperability at the 15th ECAI, Lyon, France (2002) 45-50.
13. Z. Jerroudi and J. Ziegler, iMERGE: Interactive Ontology Merging, In Proc. 16th international conference on EKAW, Italy (2008).
14. P. Bouquet, L. Serafini, S. Zanobini, S. Sceffer, Bootstrapping semantics on the web: meaning elicitation from schemas, In Proc. 15th Intl. WWW conference, (2006) 505-512.
15. F. Giunchiglia, P. Shvaiko, M. Yatskevich, S-Match: an algorithm and implementation of semantic matching, In Proc. 1st ESWS, LNCS 3053, Springer-Verlag (2004) 61-75.
16. M. Niepert, C. Meilicke, H. Stuckenschmidt, A Probabilistic-Logical Framework for Ontology Matching, AAAI (2010), 1413-1418.
17. C. Meilicke, H. Stuckenschmidt, A. Tamilin, Reasoning support for mapping revision, JLC, 19 (5) (2009) 807-829.
18. E. Jimenez-Ruiz, B. Cuenca-Grau, I. Horrocks, R. Berlanga, Ontology integration using mappings: Towards getting the right logical consequences, In Proc. 6th European semantic web conference, Heraklion, LNCS 5367, (2009) 475-479.
19. C. Trojahn and J. Euzenat, Consistency-driven Argumentation for Alignment Agreement, Ontology Matching, (2010) 37-48.
20. Y.R. Jean-Marya, E.P. Shironoshitaa, M.R. Kabuka, Ontology matching with semantic verification, Web Semantics: Science, Services and Agents on the WWW, 7 (1) (2009) 235-251.
21. M. Fahad and M.A. Qadir, M.W. Noshairwan, N., Iftakhir, DKP-OM: A Semantic Based Ontology Merger, In Proc. 3rd International Conference I-Semantics, Austria, J.UCS, (2007) 313-322.
22. M. Fahad, N. Moalla, A. Bouras, M.A. Qadir, M. Farukh, Disjoint-Knowledge Analysis and Preservation in Ontology Merging Process, 5th Intl conference on Software Engineering Advances (ICSEA), (2010) 422-428.
23. N. Maiz, M. Fahad, O. Boussaid, F. Bentayeb, Automatic Ontology Merging by Hierarchical Clustering and Inference Mechanisms, In Proc. 10th Intl. Conference on Knowledge Engineering and Knowledge Management (I-KNOW), Austria (2010) 81-93.
24. M. Fahad and M.A. Qadir, A Framework for Ontology Evaluation, 16th ICCS Supplement Proceeding, 354 (2008)149-158.