J Intell Inf Syst (2018) 51:49–70
https://doi.org/10.1007/s10844-017-0490-3

Toward analyzing impact of disjoint axioms for merging heterogeneous ontologies

Muhammad Fahad[1]

Received: 27 December 2015 / Revised: 24 October 2017 / Accepted: 27 October 2017 /
Published online: 18 November 2017

Abstract In recent years, the question on Automatic Ontology Merging (AOM) become challenging to address for the researchers. Our research and development for the Disjoint Knowledge Perservation based Automatic Ontology Merging (DKP-AOM) is a milestone in the same direction. This paper provides a more specific discussion about disjoint knowledge axioms in DKP-AOM and makes an assessment of our merge algorithm that looks-up within disjoint partitions of concept hierarchies of ontologies. The significant use of disjoint knowledge is corroborated by testing conference and vertebrate ontologies. The results reveal that disjoint knowledge axioms help identifying initial inaccurate mappings and remove ambiguity when the concept with same symbolic identifier has a different meaning in different local ontologies in the process of ontology merging. Disjoint axioms separate the knowledge in distinct chunks and enable concept matching within the boundaries of sub-hierarchies of the entire ontology concept hierarchy. While finding matches between concepts of ontologies, disjoint partitions with the disjoint knowledge about concepts in source ontologies minimize the search space and reduce the runtime complexity of ontology merging. We also discuss encouraging results obtained by our DKP-AOM system within the OAEI 2015 campaign.

Keywords Ontology merging · Heterogeneous ontologies · Disjoint axioms · Consistency and redundancy · OAEI competition 2015 · Semantic web

1 Introduction

Ontology merging is proposed as one of the solutions to achieve the demands of interoperability among multi-vendor's systems. It is a process of generating a single ontology from

✉ Muhammad Fahad
fahad.muhamamd@cstb.fr

1 Center Scientifique et Technique du Batiment (CSTB), 290 Route des Lucioles, 06904 Sophia Antipolis, France

different heterogeneous source ontologies individually working in different (or overlapping) domains. Mainly, it consists of two primary steps. First, the source ontologies are looked-up for the similarities between them. Second, duplicate-free union of source ontologies is achieved based on the established similarities. The source ontologies contain overlapping domain knowledge, but can contain different types of semantic heterogeneities which create conflicts when merged. The new merged ontology, that is the result of the union of source ontologies, should provide a unified consistent and coherent view about the source ontologies. An initial report on combining and relating ontologies with an analysis of problems and their solutions is provided by Klein (2001). During the last decade, it has been agreed by the researchers that the problem of ontology matching and merging is very hard to perform manually beyond a certain degree of complexity, size, and number of ontologies (Euzenat et al. 2007). Ontology Merging requires fully automatic methods for enabling interoperability in the dynamic environments, such as the semantic web and data warehouses where the analysis context is made on-the-fly (Maiz et al. 2010). Although, there is a great effort seen, but, still existing merging systems are semi-automatic. These systems reduce the burden of manual creation and maintenance of mappings, and also need 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. Recent studies on ontology merging show that due to conceptualization and explication mismatches between local ontologies, fully automatic merging is unattainable (Kotis et al. 2006). But, effective algorithms for computing semantic correspondences help us reach at a position, where ontology merging can be carried out with a minimum human intervention. In fully automatic merging, it is very difficult to trust on the automatic merge results without insight in the conflict resolution strategies; how elements are merged and what are the reasons behind their integration (El Jerroudi and Ziegler 2008). On the other hand, semi-automatic systems provide flexibility for determining the results of the merging process and keep the user updated about the merging sub-tasks. In addition, in a study about the gap analysis between ontology mapping and merging techniques, Anjum et al. (2010) 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". One of the methods to identify conceptualization mismatches is to analyze disjoint knowledge in the process of ontology merging (Qadir et al. 2007). Thus, in the recent ontology-based research, disjoint knowledge axioms gained much popularity as they separate the domains and build boundaries of concepts so that machine can reason in the semantically sound manner.

Völker et al. (2007) proposed learning mechanisms about disjoint knowledge within the hierarchies of concepts to automatically enrich a single ontology with disjoint axioms. Qadir and Noshairwan (2007) proved that the omission of disjoint knowledge in an ontology that serves as a backbone in the critical system may lead to erroneous and catastrophic situations, and proposed a criterion to generate alarms for the disjoint knowledge omission between concepts in ontologies. There are several intra-ontology errors that occur due to disjoint knowledge omission and wrong placement of disjoint axioms which cause inconsistency and in-conciseness in an ontology. However, these works only discuss the intra-ontology issues about disjoint knowledge within a single ontology. Besides the fact that differences in modeling can lead to conflicts (Fahad and Qadir 2008), the other hurdles for the automatic ontology merging are ontological errors (Brank et al. 2005) and design anomalies (Baumeister and Seipel 2005) present in the source ontologies. Firstly, ontological errors and design anomalies that can occur in the source ontologies detract reasoning and inference mechanisms and create bottlenecks in their integration tasks. Secondly, during merging of several on-line ontologies, we came across various situations when the individual ontologies are free from errors, but during the similarity computation, some of the identified mappings lead toward an erroneous situation producing several types of errors in the merged ontology. For building effective ontology merging algorithm, it is essential to incorporate ontological error check during the validation of the ontology mapping process. We incorporate these aspects and build semantic based ontology merging system DKP-AOM. First, our merging system DKP-AOM builds consistent and coherent mappings. Then, it merges the source ontology by established accurate mappings to build the conflict free union of source ontologies. The details of these aspects are reported in the following papers Fahad et al. (2011, 2012).

This paper provides more details about the DKP-AOM system in terms of its use of disjoint axioms. This paper describes the different steps of merging ontologies in DKP-AOM. It provides more specific discussion on disjoint knowledge axioms and corroborates these axioms with the case study results. It presents how disjoint knowledge axioms help to identify initial inaccurate mappings and remove ambiguity when concept with same symbolic identifier have different meanings in different local ontologies in the process of ontology merging. Disjoint axioms separate the knowledge in distinct chunks and enable concept matching within the boundaries of sub-hierarchies of the entire ontology concept hierarchy. While finding matches between concepts of ontologies, domain specific heuristics about disjoint knowledge between concepts in source ontologies minimize the search space and thus reduce the runtime complexity of ontology merging. Finally, it makes an assessment of our merge algorithm that looks-up within disjoint partitions and reduces search space on the basis of disjoint axioms.

The rest of paper is organized as follows. Section 2 highlights existing features of existing ontology merging systems. Section 3 defines our ontology merging process by different steps involved. Section 4 provides a case study to analyze the impact of disjoint knowledge axioms and disjoint partition lookup for the integration of heterogeneous ontologies. Section 5 concludes the paper and show future directions.

2 State-of-the-art merging systems

In the research literature, there are many diverse approaches, techniques and systems for the ontology mapping and many detailed survey papers are available for reading. In a very recent survey on ontology matching, Shvaiko and Euzenat (2013) analyzed state-of-the-art ontology matching tools and also provided future perspectives for the ontology alignment research. But their work is limited to only mapping and alignment systems and techniques, and does not incorporate ontology merging systems. They analyzed the empirical results of Ontology Alignment Evaluation Initiative (OAEI) and compared the precision and recall of systems that participated in OAEI. According to them, ontology alignment research has gained measurable progress and became much mature but with slow speed in the last decade. The reader can find their analysis of six years on ontology alignment evaluation initiative in Euzenat et al. (2011). Detailed comparison between our system DKP-AOM (Fahad 2015) and ontology matching systems can be found on OAEI 2015 official web page.[1] Here, we consider only those mapping system under the related work of this paper which have

[1] http://oaei.ontologymatching.org/2015/results/index.html

extended their work for the merging of heterogeneous ontologies. Therefore in this section, we discuss approaches and systems only for the ontology merging.

Stumme and Maedche (2001) proposed a Formal Concept Analysis methodology called as FCA-MERGE based on a bottom-up technique for the ontology merging. It takes the source ontologies and text documents which contain instances, and matches the application specific instances. When it finds common instances between them, then the concepts that instantiate them are taken as candidates for merging. Finally, it adopts the semi-automatic methodology with user intervention to build the target merged ontology. The instance based approach serves best when instance repository is available, but, this is not the case for on-line ontologies which are mostly without instance repositories. Noy and Musen (2003) developed the interactive ontology merging tools named iPROMPT, i.e., Prompt equipped with Anchor-Prompt. Prompt, a semi-automatic tool, exploits label matching techniques for the initial comparisons between source ontologies to produce suggestions to the end-users who then merge heterogeneous ontologies interactively. When the user accepts a suggestion, Prompt system merges the concepts and updates the suggestion list, and this process goes iteratively until the merged ontology is achieved. During the merging process, it identifies name conflicts and redundancy in the class hierarchy, and suggests user to take actions. In Anchor-Prompt, they embedded graph based techniques to identify more similarities between concepts of source ontologies. The PromptDiff tool compares two versions of the same ontology and finds structural differences between them. PromptFactor aims at generating newer ontology from an existing ontology. It allows user to extract the desired portions of large ontology into a new ontology preserving the semantics of the concepts. These tools are available as plug-ins of Protégé which is the mostly used open source ontology editor by the semantic web community. McGuinness et al. (2000) developed Chimera for the ontology editing, merging, and diagnostic environment to meet the demands of representation and reasoning tasks on the Web. They use concept label matching techniques and terminological resources (such as term names and definitions, possible acronym, suffix similarity and expanded forms) for the merging of ontologies. Like Prompt, Chimera performs merge operations on found matchings between the source ontologies on the user feedback. These are the pioneer ontology merging tools, but today's rich ontologies need more than this functionality for the merging of complex semantic web OWL ontologies.

The approach proposed by Chalupsky (2000) is not to produce a complete merged ontology, but, a bridge ontology to provide services, such as dataset translation, ontology extension generation and querying different ontologies. The list of suggestions containing candidate merge concepts is identified and provided to the end-user. The drawback appears as an end-user receives no guidance during the merging process of heterogeneous ontologies except the initial list of matches. Mitra and Wiederhold (2002) proposed ONION System, which is different from other systems, rely on producing a separate single merged ontology from the existing ones. ONION system produces rules/articulations between the source ontologies by using the string matching, structure of taxonomy, co-occurrence of words in a text corpus for the merging of heterogeneous ontologies. The rules or linkages between the source ontologies are generated by a semi-automatic tool with the help of an end-user. This system is designed especially for promoting interoperability between heterogeneous sources to reliably answer user queries. Kim et al. (2005) developed many facilities such as importing, matching, modifying and merging of ontologies in their tool named MoA. Their tool uses string matching and synonym matching by associating concepts with their meaning with the help of Wordnet for finding correspondences between the source ontologies. They also focus to handle compound words within the source ontologies. Its intermediate output is related to ONION as it produces articulation rules, and finally produces a new merged ontology like other semi-automatic systems (Prompt and Chimera).

Kotis et al. (2006) proposed the HCONE-Merge methodology which uses linguistic and structural knowledge about source ontologies to formalize the Latent Semantics Indexing method. Then, they use the Latent Semantic Indexing mechanisms for the computation of all possible mappings by analyzing intended informal meanings of source concepts with the Wordnet senses. Then the merging process needs human expert for the validation of the proposed mappings between the source ontologies. Finally, they use the reasoning services of description logic for the automatic merging of local ontologies. In their methodology, WordNet plays an important role, therefore HCONE-Merge gives poor performance without exploiting it. There is another way of merging ontologies based on prioritized input ontology named asymmetric merging also known as target-driven merging. Raunich and Rahm (2011) developed a tool named ATOM for the automatic target-driven ontology merging. ATOM merges large taxonomies based on the Equivalence Matching (EM) strategy between a source and target taxonomy. Their target-driven methodology saves the structure of the target ontology (prioritized input ontology) and instances of both input ontologies during the merging of source taxonomy into the target taxonomy to produce a unique merged ontology. Their asymmetric nature of merging algorithm is helpful when extending an ontology with the additional source ontologies. They integrated their methodology with the COMA++ schema matching system which allows semi-automatic execution for the merging process. Their methodology is only limited for the extension of an ontology and does not feasible for the merging of ontologies having complex DL axioms or terminological differences.

LogMap toward ontology mapping (Jiménez-Ruiz and Grau 2011) and ContentMap toward ontology merging (Jiménez-Ruiz et al. 2008) are the significant milestones toward ontology based research. ContentMap tool provides a general method to facilitate the integration of heterogeneous ontologies by using existing mapping tools (such as OLA, CIDER, AROMA). For the integration of ontologies, the user has to provide initial mappings or existing mapping tools can be used for identifying mappings, evaluate the entailments before and after the integration and repair the consequences of the integration process. They also extended their work and provided a logic-based methodology (2009) to support the user in the safe use of imported symbols and in the economic import of the relevant part of the imported ontology. Their tool named Safe Protégé Manager (ProSE) as a protégé plug-in supports safe import and relevant parts of the imported ontology. (See comparison between LogMap family tools and our DKP-AOM on the official site of OAEI). El Jerroudi and Ziegler (2008) developed a tool for the semi-automatic merging of ontologies named IMerge which provides different visualization facilities. First, the SmartTree-View displays the hierarchical structure of source ontologies. Second, the Matrix-View compares two source ontologies and provides visualization of the possible mappings between them.Third, the Merge-View builds an interactive mode of merging with the user feedback by accepting, changing or rejecting proposed options for each of the candidate Merge options. The candidate pair concepts for the merging are detected based on the string, structure and additional input document that has annotations to the concepts of source ontologies. The drawback appears when the source ontologies are much complex then only consisting hierarchical structure. Another approach for the automatic ontology merging is formalized based on the hierarchical clustering and inference mechanisms (Maiz et al. 2010). They proposed hierarchical clustering algorithm for the computation of equivalent entities between two or more ontologies and use description logic services of Pellet Reasoner to find all possible relations between the concepts of source ontologies. Finally, it integrates the correspondences between source ontologies to build the automatic global merged ontology for the mediation based virtual data warehousing. The drawback of their approach is that it ignores the underlying semantic inconsistencies which possibly arise for the generation of merged ontology.

The idea of using disjoint axioms for logic based reasoning seems to be rather obvious, but there is not much work devoted to it in the research literature. Most importantly, the above systems do not consider disjoint axioms in building search space and analyzing inconsistent mappings between the ontologies. Therefore, we took this challenge and start analyzing whether the disjoint knowledge between concepts is useful and how much helpful in reducing time and space complexity for the merging of heterogeneous ontologies. We conclude that the analysis and use of disjoint knowledge during the mapping and merging of ontologies reveals consistent mappings in the Query Answering (OA4QA) track at the ontology alignment evaluation initiative (OAEI).

3 Ontology merging process

From the state-of-the-art ontology merging system, we analyze that the merging process is dependent on the human expert for the validation of mappings and resolution of conflicts. In addition, we analyze that there is a need of fully automatic methodology of ontology merging for enabling interoperability in the dynamic environment, such as the semantic web and virtual data warehouse where analysis context is required on-the-fly. The contribution presented in this paper minimizes human involvement during the ontology merging process by the automatic detection of semantic inconsistencies in the initial stages of ontology merging and applying automatic conflict resolution strategies for building a merged ontology without human involvement. We analyze all types of information, especially disjoint knowledge analysis and preservation in the 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 a 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 the local ontologies. It checks whether the lexically same concepts within the local source ontologies must not contradict with each other by the 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. In the case of various conflicts, it exploits various resolution strategies automatically and builds the merged ontology without any human involvement. Due to the use of more semantics present in the source ontologies, test criteria for the validation of mappings and automatic conflict resolution strategies, our approach enhances the accuracy of mappings and produces consistent, complete and coherent global ontology. We aimed at managing the quality of mapping and merging, also, with the intention of reducing its run-time complexity. Our hypothesis is that the mapping lookup under disjoint partitions (or subclass hierarchies) at the first time can generate most of the candidate mappings. By a divide and conquer strategy based on disjoint partition look-up, ontology merging process can gain tremendous efficiency, which is desired for the large practical ontologies. Since, our solution for ontology merging performs disjoint knowledge analysis and preservation, and the mapping lookup is also based on Disjoint Knowledge Partitions, we named it as DKP Automatic Ontology Merger (AOM). The detail about the various aspects of our ontology merging framework, i.e., merging process, algorithm, conflict resolution, etc., is discussed as follows. Our ontology merging approach is based on the following steps of the process model. These steps are explained below for both modes of merging operation, i.e., semi-automatic and fully-automatic. We also compare our merging process with the merging process of semi-automatic ontology merging system Prompt (Noy and Musen 2003).

3.1 Feature engineering

The ontology merging framework gets OWL ontologies for their merging purpose. At the initial step, it has to perform certain tasks such that merging can be performed easily. These tasks are the formulation of ontology graphs and pre-processing of concept labels.

Formulation of OWL graphs. One of such tasks that facilitates ontology matching and merging is to formulate the OWL ontologies in memory in such a way that underlying processing can be performed easily. Jena Semantic Web Framework aims at providing these services for the semantic ontologies. As compared with Prompt that used RDFS, we formulate OWL ontology graphs based on the Jena semantic web framework.

Pre-processing of Ontology Terms. Before comparing labels/terms of concepts and properties in ontologies, it is vital to perform normalization, tokenization and elimination as a pre-processing step. It significantly improves the results of the matching process because of inherent semantic heterogeneities of terminologies used in source ontologies. Secondly, when performing synonym based matching, the dictionary or lexical database can recognize them. There are many types of normalization and tokenization that can be employed to gain advantages. These are as follows.

1. Case Normalization. Case normalization is a process in which all the characters in the concept name are transformed in either lowercase or uppercase format before comparison. For example, ‘UNIVERSITY’ or ‘University’ is changed to ‘university’.
2. Special Character/Blank/Digit Stripping. Concept names that contain special symbols like underscore, apostrophes, dashes, etc., are stripped before comparison. For example, ‘MS-Student’ is split into ‘MS’ and ‘Student’ terms. Similarly, digits and whitespaces like blank, spaces, tabulation, etc., are also trimmed. For example, ‘MacDonald4U’ is transformed into two labels ‘MacDonald’ and ‘U’.
3. And Stripping. ‘And’ or ‘&’ are stripped off from the concept name. For example, concept names ‘Photo and Camera’ or ‘Camera & Photo’, return a set of two elements, i.e., Photo, Camera
4. Tokenization. Compound words need tokenization so that when performed synonym matching, the dictionary or lexical database can recognize them. For example, when concept ‘CsDepartment’ or ‘PhdStudent’ are looked in the Wordnet for synonyms, no results are retrieved. Tokenization of these words into Cs, Department and Phd, Student would be beneficial and provide proper senses of these concept terms. It is significant in the case of concepts and properties that are represented in the source ontologies by the joint (or compound) words. Without tokenization, identification of these concepts and properties is not possible. This results low precision and recall values, and also creates redundancies in the merged ontology. For example consider property pairs, such as (displayManuscript, showManuscript), (giveDemo, presentDemonstration), (recordPresentation, tapePresentation), (makeSchedule, create-Agenda), (ReviewAndSuggest, critiques_and_proposes), etc. Identification of such properties are important and challenging task, but is only possible if we apply tokenization mechanism. It is also

important to apply Elimination technique to remove special and stopwords (for instance And, _, -, etc.) before tokenization technique, otherwise WordNet does not recognize the substrings and we cannot achieve our purpose. Figure 1 shows an algorithm that DKPAOM embeds for the tokenization of semantically similar properties. First, it calls the method named Joint_Synonym_Properties to get the splitting index of the compound-word property which is based on the method getSplittingIndex. Method getSplittingIndex gets substrings c and d i.e., c(0,i) and d(i,n) from the start of the compound word till its ending. Then it checks their existence in the wordnet for the substrings c and d, returns the index when both c and d are recognized in the WordNet. On getting the index, method Joint_Synonym_Properties makes the tokens of compound word. Finally, it detects mappings of tokens and proposes their mappings. At the time of OAEI 2015 participation, the proposed algorithm for the tokenization of ontological entities only tackles joint words that contain two individual words (e.g., make_Schedule). Due to the semantic heterogeneities it is possible that web ontologies contain joint words that contain more than two individual words. But, due to

```
Joint_Synonym_Properties( String word1 , String word2)
{
        int index1 = get an index to split the joint word1        // See spliting index function below
        int index2 = get an index to split the joint word2

        String subC1 = get substring of word1 from index(0, index1)
        String subC2 = get substring of word2 from index (1+index1 , end )
        String subD1 = get substring of word1 from index(0, index2)
        String subD2 = get substring of word2 from index (1+index2 , end )


    if(synonymMatch(subC1, subD1)==1 &&synonymMatch(subC2, subD2)==1 )
            print: tokenization is successful and properties are semantically similar
            else if( synonymMatch(subC1, subD2)==1 &&synonymMatch(subC2, subD1)==1 )
                    print: tokenization is successful and properties are semantically similar
             else         print: properties are different
}

int getSplittingIndex (String sa) {        // returns an index to split sa into two substrings

    char []ch = get char array of string sa
    for(int i=1; i<ch.length; i++){
            string c = get substring of sa from index ( 0, i )
            string d = get substring of sa from index ( i+1, end )

            if(checkExistance_InWordNet(c,d) ==1 )
                    return i;         // if both words are recognized by the wordnet
                    }
    return 0;
}
```

Fig. 1 Algorithm of Tokenization technique embedded in DKP-AOM

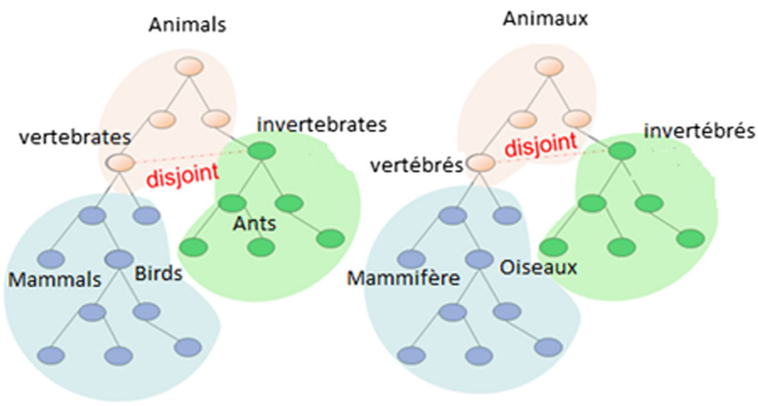

Fig. 2 Disjoint partitions in Vertebrates ontologies

the complexity of checking substrings in the wordnet, we restricted our algorithm for checking only joint words that comprise two separate words.

5. Elimination. Often property names use prepositions, articles or helping words to add semantic expressions between the concept names. For example, RegistersIn (Student, Course), ReviewedBy (Paper, Reviewer), has Address (Person, Location), etc. Therefore, properties need elimination of helping words such as 'In', 'By', 'From', 'Has', 'A', 'An', 'The', etc.

3.2 Identification of mappings

The desired task of ontology merging requires the identification of merging candidates on the basis of mappings between the constructs of ontologies. Identification of mappings is based on these steps.

1. Selection of Search Space: This is very important to build the search space for the lookup of mappings between ontologies. In general, it requires exhaustive analysis (or complete comparison) for the similarity computation between the concepts of ontologies, where each concept c of the ontology Oa is matched with each concept c' of the ontology Ob. This is similar to the existing methodologies of ontology merging systems, e.g., Prompt. But, we propose a divide and conquer approach to build up the search space with the help of disjoint partitions. For example, consider Vertebrates ontology in Fig. 2, where disjoint axiom partition the vertebrates into two disjoint partitions, i.e., Birds and Animals. We are using microsoft translate API[2] to find mappings between multi-lingual ontologies. When our algorithm gets mappings between concepts (vertebrates and vértébrés), and (invertebrates and invértébrés), it looks the mappings of their children concepts under their hierarchy and not in the disjoint partition or in the whole ontology. This restriction minimizes the search space for the mapping look up. We believe this idea of divide and conquer is very helpful for large ontologies. The more disjoint axioms that are modeled in ontology, the more the search

[2] Microsoft Translate API: https://code.google.com/p/microsoft-translator-java-api/

space is restricted, which impacts in lower the number of comparisons for the identification of mappings. For example, consider a situation having primary mapping between (O1_Animal : O2_Animaux), in addition there are many types of Dogs within the hierarchy of concept Animals only, and made disjoint with other categories of animals. In such a situation, when concept Dog is mapped on the concept Chien (in french), then the children of Dog concept should be searched under the subconcepts of Chien. But, It may be possible that there is no concept 'Chien' where O1:Dog should be mapped on O2, and ontologist has placed all the types of Chien under Animaux concept. Therefore, this requires searching in the same level Animaux in O2. Our algorithm builds the search space on the basis of the same criteria of disjointness, and priorities the search space. First, it looks for the mapping concepts within the search space under disjoint axiom. If the mapping is not found, then it looks for the upper level space of disjoint concepts, as it represents the more general space. The least priority is given to the disjoint sibling space for the mapping look up.

2. Similarity Computation: Heterogeneous ontologies need various types of similarity measures for the identification of mapping candidates. Prompt uses string matching for finding only the identical labels. But, we have seen in the previous sections, that only string matching technique does not tackle all the situation of semantic heterogeneities. Therefore, our similarity computation is based on many parameters (syntactic, semantic, axiomatic, etc.). Each of the parameters has its own value, and this value may vary according to the ontologists' perception. The utmost value is for the label of concepts, but, the label itself only does not correspond to the semantics of the concept. Thus, weighted mechanism is adjusted that defines the value of these parameters during similarity computation.
3. Similarity Aggregation: 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. Prompt exploits only one similarity measure that does not require aggregation. Our methodology exploits many similarity computation factors, therefore aggregation is required to find the best possible mapping candidates. 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 the enumerated class ContributionType in the source ontologies O1:ContributionType oneOF AbstractPaper, PositionPaper, ConferenceFullPaper, O2:ContributionType oneOF AbstractPaper, ConferenceFullPaper. A string-based Label matcher Sim(lab) produces a similarity value equal to 1, as their labels 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 as below. Sim(lab) = 1 and $sim(axm) = s_1 \cap s_2 / s_1 \cup s_2$ Sim(axm) = $2/3 = 0.667$ For the aggregation of similarity values, the maximum value (i.e., 1) is chosen as the best representative similarity value between ContributionType 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. Let there are m number of matchers. For a concept con in the ontologies Oa and Ob, each matcher produces its output as Sim(mch) that denotes the similarity value between the concept con in the ontologies

Oa and Ob. Then the aggregated similarity value Sim(con) is the average value computed by summing the individual similarities Sim(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 $(1+0.667)/2 = 0.833$. 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 this 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, a weighted average is calculated as: $weighted\_mean = \frac{w_1x_1+w_2x_2+...+w_nx_n}{w_1+w_2+...+w_n}$

On the basis of this formula, weighted aggregated similarity value Wg_Sim(con) is computed by multiplying the weight wgi to the Sim(mch) value before taking their average. As an input let the user give weight = 1 to the label matcher and weight = 0.8 to the axiom matcher, then the Wg_Sim(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, 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 Wg_Sim(con) (weighted average) is computed when there are m matchers each with weight Wg.

$$wg\_sim(con) = \frac{\sum_{i=1}^{m} \mathrm{wg_i} * \mathrm{sim_{mch_i}}}{\sum_{i=1}^{m} \mathrm{wg_i}} \tag{1}$$

4. Interpretation When the candidate pairs based on different similarity measures are computed, there is a need to define interpretation for the best candidate pairs of mappings which lead toward the generation of a merged ontology. One criterion is to define a threshold value as in the case of Prompt. The candidate pairs with a similarity value above a defined threshold are considered as merge candidates, and presented to the user as a list of suggestions for their merger. In our merge methodology, validation is the fundamental part to make interpretations about the candidate merge pairs. Hence, interpretation is based on the validation of mappings and only validated mappings serve as candidates for merging.

3.3 Validation of mapping

Validation of mapping is necessary for achieving the consistency and accuracy of a merged ontology. There are many factors on the basis of which, the merging process can validate the mappings so that the merged ontology stays consistent, complete and coherent.

3.4 Merging of candidate mappings

Merge Operation requires the list of validated mappings so that each candidate mappings are merged together. There are two main considerations in this step, i.e., Conflict Resolution mechanism and Execution of Merge Operation. In semi-automatic mode, resolution of conflict and selection of merge candidate is based on the user feedback. In Prompt, for every conflict or mismatch, the system prompts to the user to handle the situation. Then, with the subject knowledge and intelligence of user, candidate pair are selected from the list of suggestions and merged to produce the merged ontology. In our methodology, where there is a preference given to the source ontology, most of the conflicts are solved as merger has to follow the semantics of preferred ontology. For example, consider the case where a property Amount:float in Oa and Amount:Double in the preferred ontology Ob are going to be merged. The merging process automatically resolves such conflict on the basis of preference. In another case, where preference is not given, it looks up in the resolution table or does apply inference mechanisms for the resolution of conflicts during the merge operation. Finally, our merging process is executed as to produce a merged ontology. Our algorithms for the merging of Axiomatic definitions of concepts are further elaborated here (Fahad 2017). We believe the execution of ontology merging process should be executed in such a manner that reduces costs of machine resource, i.e., memory and time. Our merging algorithm generates the merged ontology by merging a single source ontology into another. First, it performs merging of level one concept, and then map and merge the rest of the concepts below them (i.e., sub-hierarchies). There are many reasons behind the execution of our merged algorithm in this manner. Firstly, it is easy to get all the knowledge of O1 and add O2 into it to generate a merged ontology O3 which unifies and conceptualizes all the domain knowledge of source ontologies. Secondly, concepts at level 1 are more generic than concepts at other levels and can be used in defining contexts of other concepts, therefore we believe merge algorithm should first look for their mapping and perform merging. Once level one concept are mapped and merged, then the rest of the concepts are looked up and merged under their parent concepts. The algorithm is presented in Fig. 3 and the evolution of O3 as a merged ontology is depicted in the Fig. 4.

Input:

1. The source ontologies O1 and O2
2. Preference known of O1, if unknown consider O1 as default
3. The*MappingList* sorted by level of ontology O2

Output: Merged Ontology O3

Algorithm:

For the execution of an algorithm, consider concept c belongs to O1 and concept d belongs to O2.

1 Make a copy of the O1 ontology to merge ontology O3.
2 Initialize Queue A with the level 1 concepts of O2.
3 Do steps 4 and 5 on all the concepts of Queue A, until it is empty.
4 Get a concept c from the Queue A, look its mapping concept from the mapping list.
- If there is a mapping concept d, then
  - Look for conflict resolution if there is a mismatch
  - Merge concepts c and d in O3.
  - Merge associated properties and axioms
- If there is no mapping for the concept c,
  - If c is level=1 concept, then simply add it in the O3 under 'Thing'
  - If c is level? 1 concept, then simply add it in the O3 under its Parent concept

5 Add all children of c to A (i.e., start performing step 4 on all the children of concept c)
6 Finish the procedure, when A is empty.
Ontology O3 is the merged ontology.

Fig. 3 Execution of ontology merging process

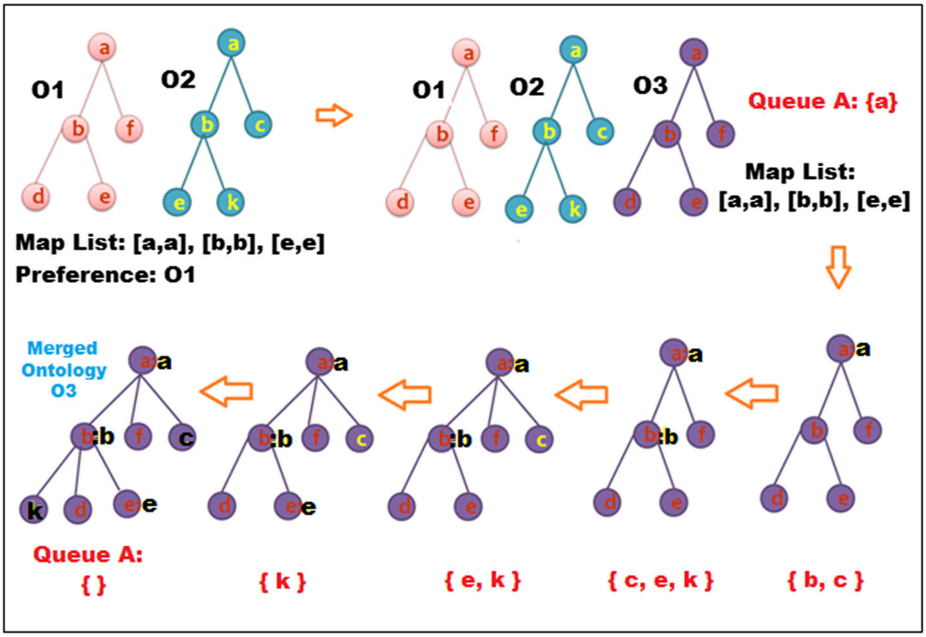

Fig. 4 Evolution of O3 as a merged ontology

3.5 Verification of merged ontology

Once the merged Ontology is generated, it is highly important to check its accuracy. It may be possible that the merged ontology contains inconsistencies, incompleteness and/or redundancies. Existing ontology DL reasoning or evaluation tools can be used to check the verification of a merged ontology. We build our evaluation criteria based on Consistency, Completeness and Redundancy criteria for the verification of merged ontology and this is well elaborated in Fahad et al. (2012), and out of scope of this article.

3.6 Iteration

The process of merged ontology generation has a very high complexity. The whole process from mapping identification to their validation and conflict resolution to merged ontology generation can be error prone. In addition, for semi-automatic mode of ontology merging, it is necessary to perform iteration to allow manual refinements in the underlying steps. When the user selects the candidate concepts for merging, merged concept is generated in the merged ontology, and the system performs an iteration to see the impact of merged concepts, verifies its consistency and accuracy, and updates the list of suggestions for next candidate pairs for merging. Fully automatic mode of ontology merging also requires iteration to check and analyze the accuracy of merged ontology. It verifies that the merged ontology unifies all the information present in the source ontologies in a consistent and concise manner.

4 Assessing the importance of disjoint axioms and disjoint partition lookup

For finding the best match of any concept of ontology O1 with n1 concepts needs an exhaustive analysis with entire concepts of ontology O2 with n2 concepts, resulting n1xn2 comparisons between them. Disjoint axioms separate the knowledge in distinct chunks and enable concept matching within boundaries of sub-hierarchies of the entire ontology concept hierarchy. When disjoint axioms are properly placed in class hierarchies, then the run-time significantly reduces. For example, in Vertebrates and Vértébrés ontologies, when Birds are mappings on Oiseaux concept and Animal on Animaux, specified that these concepts are disjoint in individual ontologies, then the merging algorithm should look the mappings of sub-concepts of Birds under the sub-concepts of the Oiseaux concept and not in the whole ontology. This concept of lookup within disjoint partitions significantly reduces the search space of the merge algorithm. We have tested the multilingual ontologies with and without disjoint axioms to measure the efficiency of the merge algorithm. The test is performed on 4 sets of ontologies and the runtime complexity is measured.

1. Test set 1 contains Vertebrates ontology (in English) with 14 concepts and Wirbeltiere ontology (in German) with 15 concepts. The class hierarchies of these ontologies (Vertebrates, Wirbeltiere).
2. Test set 2 contains CMT Conference ontology with 36 concepts and CRS_DR with 14 concepts. These ontologies are well known about the conference domain and developed for the conference management systems. The detail of these ontologies is provided in section 4.3.
3. Test set 3 contains Vertebrates ontology (in English) with 50 concepts and Vértébrés ontology (in French) with 52 concepts.
4. Test set 4 contains Vertebrates ontology (in English) with 85 concepts and Vértébrés ontology (in French) with 86 concepts.

4.1 Estimated time calculations

On the basis of these data sets, time is measured to access the importance of disjoint axioms in the source ontologies. Figure 5 shows the statistics of the experiment performed with

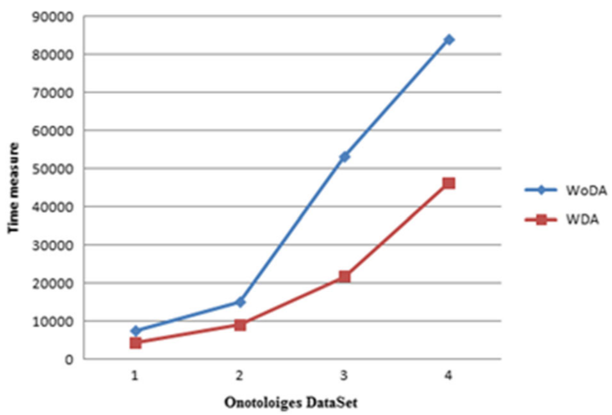

Fig. 5 Estimated time consumed with and without disjoint axioms lookup in ontologies

disjoint axioms (WDA) and without disjoint axioms (WODA) approach for the mapping lookup. From the estimated time, we conclude that the disjoint partition lookup strategy works well for building the search space for the mapping identification between the heterogeneous ontologies. Once the upper level concepts are mapped, then the lookup for their children should first be made in their down hierarchies, rather than in all the ontology. This technique of lookup serves good in reducing time complexity and memory resources, especially when source ontologies have a large number of concepts.

4.2 Comparison performed for concept mappings

In O1:Vertebrates and O2:Wirbeltiere ontologies, disjoint axioms between level-1 concepts partition the domain concepts into two non-overlapping domains. There are 14 concepts in O1:Vertebrates and 15 concepts in O2:Wirbeltiere. For concept matching, it needs $14 * 15 = 210$ comparisons. But, the lookup in disjoint partitions makes the search space much smaller, and requires maximum 71 comparisons for mapping the concepts of these ontologies. In the conference ontologies, there are 14 concepts in O1:CRS_DR and 36 concepts in O2:CMT. In CRS_DR ontology, three disjoint axioms between level-1 concepts, partition the domain concepts in four non-overlapping domains, i.e., Program, Person, Document and Event. CMT ontology partitions the concept into six disjoint categories, i.e., Person, Decision, Document, Conference, Preference, etc. Figure 6 shows the top level disjoint partitions in CRS_DR and CMT conference ontology. The ontologies allow the concept mapping search space look-up within disjoint partitions. For example, search spaces look-up within (O1:Person, O2:Person), (O1:Document, O2:Document), (O1:Event, O2:Conference). For concept matching, it needs $14 * 36 = 504$ comparisons. But, the lookup in disjoint partitions makes the search space much smaller, and requires maximum 155 comparisons for mapping the concepts of these ontologies. From this experiment, we conclude that Disjoint Axioms Analysis plays a vital role in controlling the search space for finding similarities between ontologies. Look up within disjoint partitions significantly reduces the time complexity of the merging algorithm. While finding matches between concepts of ontologies, domain specific heuristics about disjoint knowledge about concepts in source ontologies minimize the search space and thus reduce the runtime complexity of ontology merging.

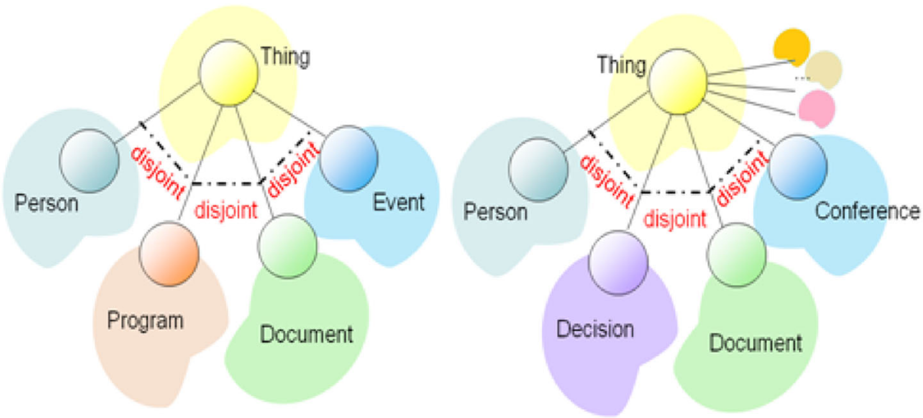

Fig. 6 Top level disjoint partitions in CRS_DR and CMT Conference Ontologies

Table 1 DKP-AOM results on conference track ontologies

Alignments	Precision	F.5-measure	F1-measure	F2-measure	Recall
ra1-M1	0.84	0.77	0.69	0.63	0.59
ra1-M3	0.84	0.74	0.63	0.54	0.5
ra2-M1	0.79	0.72	0.63	0.57	0.53
ra2-M3	0.79	0.69	0.57	0.49	0.45
rar2-M1	0.78	0.72	0.65	0.58	0.55
rar2-M3	0.78	0.69	0.59	0.51	0.47

However, domain specific heuristics can only be applied on well-known ontologies where we do not expect any alignment conflict within the sub-hierarchies of disjoint concepts.

4.3 Merging conference ontologies

Due to unavailability of merging systems, we are unable to conduct an experimental comparative analysis between them. Therefore, we participated in the OAEI 2015 in order to show the efficiency and effectiveness of our system. Our participation results are found here Conference,[3] OA4QA[4] and Anatomy[5] 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. DKP-AOM is among five ontology matchers whose alignments allowed to **answer all the queries** of the evaluation. The best global results have been achieved for violation queries, that has 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. 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.

Here, we are elaborating results of our system on the Conference domain ontologies. The goal of conference track is to find alignments among 16 ontologies[6] relatively smaller in size (between 14 and 140 entities) but rich in semantic heterogeneities about the conference organization domain. As a result, 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. 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). We achieved F-Measure values better than the two Baselines results (edna, StringEquiv). Table 1 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 toward detecting heterogeneities for the goal of ontology matching.

[3]Conference track: http://oaei.ontologymatching.org/2015/conference/eval.html

[4]OA4QA track: http://www.cs.ox.ac.uk/isg/projects/Optique/oaei/oa4qa/2015/results.html

[5]Anatomy track: http://oaei.ontologymatching.org/2015/results/anatomy/index.html

[6]Conference ontologies: http://oaei.ontologymatching.org/2015/conference/index.html

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. While 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 ontologies (Solimando et al.). Our DKP-AOM is among the five best tools which have no consistency principle violation (see Table 2 bold entries), as we have employed various algorithms for the validation of initial mappings. For evaluation details, please refer to oaei evaluation results. The lowest number of conservativity principle violations has LogMap-C which has a repair technique for them. DKP-AOM has produced second-lowest number of conservativity principle violations, and employed algorithms to maintain conciseness and avoid redundancies in the resultant ontology. Conservativity principle violations can be favored 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.

We have also evaluated our research prototype implementation for ontology merging and performed different experiments on the web ontologies that belong to the 5 different categories, i.e., University, Publication, Book, Travel and Conference. Here, we are elaborating only the experimental results about the Conference ontologies developed in OWL by different communities. The Conference ontologies are PCS, CRS_DR and CMT, which are freely available on the internet and being used by Ontology Alignment Evaluation Initiation (OAEI) since many years. CMT ontology is used for the Conference Management Toolkit (CMT) developed by the Microsoft's Academic Conference Management Service. It is written in ALCIN(D) with concept (36), datatype properties(10), object properties (49) and has 27 dijsoint axioms. CRS_DR ontology is developed by the Conference Reviewing System (CRS) to manage scientific research paper submission and reviewing. It is written in ALCIF(D) with concept (14), datatype properties(2), object properties (15) and has 12 disjoint axioms. PCS ontology is developed by the Precision Conference Solutions (PCS). It is written in ALCIF(D) with concept (23), datatype properties(14), object properties (24).

Analysis and discussion One can download subset of all alignments for which there is a reference alignment generated by the SEALS platform automatically for OAEI 2015 . But, in this subsection, we only limit our experimental details for ontology merging. We have manually checked the OWL ontology constructs in global merged ontology to measure the consistency, completeness and coherency. The experiment result can be positive or negative depending on the merging algorithm's accuracy. Figure 7 illustrates an output ontology O3

Table 2 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	0	27	1.29	7	0.33
LogMap	0	21	0	29	1.38	0	0
AML	0	21	0	39	1.86	0	0

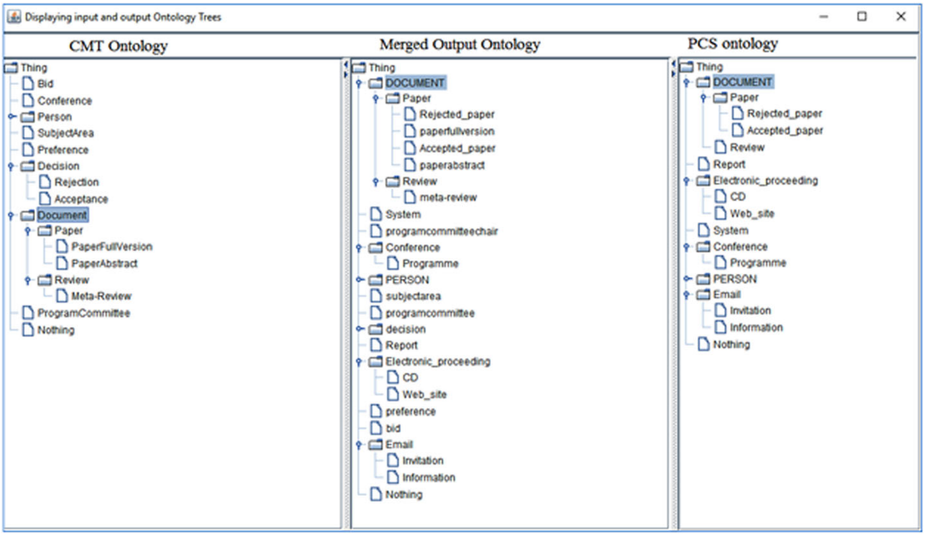

Fig. 7 Output ontology O3 from merging CMT and PCS Conference Ontologies

from merging CMT and PCS Conference Ontologies. The experimental results for each concept may or may not match with the manual expert discussion, resulting in four different cases.

1. First, True Positive (TP) or Correct Mappings, which means that OWL Concept Ca is correctly mapped on Cb in merged ontology and Human Expert is agreed with it.
2. Second, True Negatives (TN) or Correct Not-Mappings, which means that OWL Concept Ca is not mapped on Cb in merged ontology and Human Expert is agreed with it.
3. Third, False Positives (FP) or Incorrect Mappings, which means that OWL Concept Ca is incorrectly mapped on Cb in merged ontology and Human Expert is not agreed with it.
4. Fourth, False negative (FN) or Missed Mappings, which means that OWL Concept Ca is not mapped on Cb in merged ontology and according to Human Expert it should be mapped. On the basis of these four cases, Precision and Recall values are calculated with the following formula $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$, and visualized in Fig. 8.

The overall results of our hybrid approach are promising as the test criterion has rejected the initial incorrect linguistic mappings. The experiments show that our approach with automatic inconsistency detection yields a significantly higher precision. 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 make. Working with conference ontologies yield precision equal to 1, which is very good for automatic ontology merging. 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

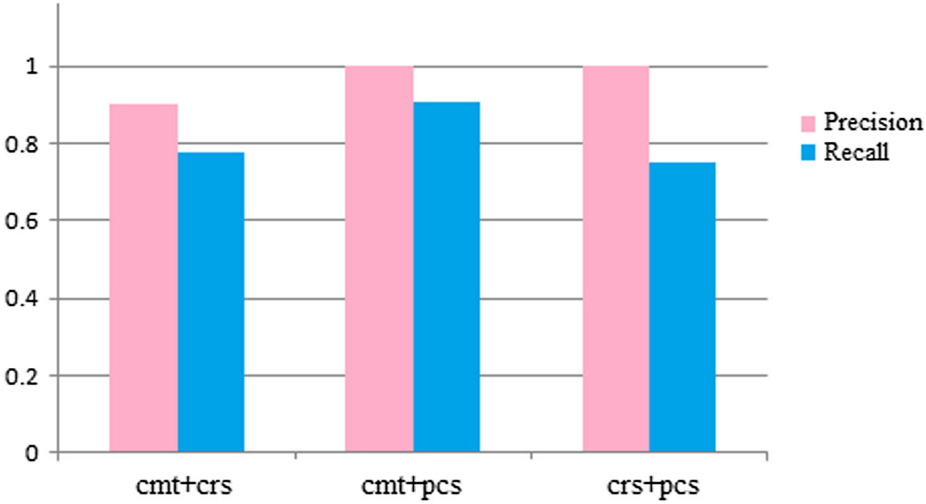

Fig. 8 Precision and recall on conference ontologies

it should make according to the human expert. Inconsistent situations in the class hierarchies are very much common dealing with the real world heterogeneous ontologies due to inherent semantic conflicts that came out from different communities over the web. These inconsistencies between local ontologies can also occur during the mappings of properties to build the property hierarchies in the merged ontology, but with lesser tendency. Our hybrid merging approach exploits basic techniques (string and synonym matching), analysis of axiomatic definitions and disjoint knowledge between concepts and properties during merging of local ontologies, and applies quality criteria to avoid erroneous situations in global merged ontology. In this way, ontology mapping and merging system finds more reliable similarities between concepts and properties between heterogeneous ontologies, and enable them to communicate, share, and exchange information in semantically sound and consistent way in the presence of several types of semantic heterogeneities. We have implemented the algorithm of detecting inconsistencies and evaluated the working of the system on the real world anthologies. 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. Our experiments on the real world ontologies yield good precision and recall values. The hybrid approach employs a restricted criterion that produces significant higher values for precision, as it aims at identifying real mappings. But, such restricted criteria often ignore some of the mappings, which effect recall values. By manual inspection, we conclude that the incorrect or missed mappings are due to the concept labels formed from composite words or defined by technical terms, and their detection is beyond the power of linguistic or synonym strategies. By these experiments, we conclude that there are several benefits of disjoint knowledge analysis and preservation in ontology merging, as outlined below.

1. Incompleteness in Merged Ontology: According to the Ontological Error's Taxonomy, disjoint knowledge omission among concepts in ontology is categorized as Incomplete Partition Error (Fahad and Qadir 2008). During the ontology merging process, if

we ignore disjoint knowledge axioms then the merged ontology would be incomplete with respect to all type of knowledge hidden in local ontologies leading to catastrophic complications in practice.

2. Inconsistency in Merged Ontology: Disjoint knowledge analysis avoids chances of inconsistencies in merged ontology. When disjoint knowledge is not considered in the ontology merging process, initial mappings lead toward inconsistent merged ontology with respect to local ontologies. For example, let O1 and O2 be two local ontologies. In ontology O1, concepts Student and Employee are taken as disjoint which means that there is no instance student that is also an instance of employee. But, in ontology O2, Student can be an Employee, e.g., PhD Researcher, Erasmus Mundus Scholar, etc., and hence represented as overlapping concepts. When these local ontologies are merged, then a common class (i.e., PhD_Researcher) between disjoint classes (Employee and Student) occurs which creates inconsistency in the global ontology.
3. Reduce Search Space and Runtime Complexity: Disjoint axioms separate the knowledge in distinct chunks and enable concept matching within boundaries of sub-hierarchies of the entire ontology concept hierarchy. While finding matches between concepts of ontologies, domain specific heuristics about disjoint knowledge about concepts in source ontologies minimize the search space and thus reduce the runtime complexity of ontology merging. However, domain specific heuristics can only be applied to well-known ontologies where we do not expect any alignment conflict within the sub-hierarchies of disjoint concepts. This point is well elaborated above by testing on real world ontologies.
4. Detecting Inaccurate Mappings by Concept name matching strategy: Disjoint knowledge axioms help identifying initial inaccurate mappings and remove ambiguity when concept with same symbolic identifier have different meaning in different local ontologies in the process of ontology merging. For example, consider ontologies O1 and O2. In ontology O1, the Course concept is further classified as BS and MS courses, and in ontology O2 the concept Student is categorized according to his qualification as BS, MS and PhD students. If mapping and merging based on concept label maps BS of O1 to BS of O2, based by linguistic (or synonym) matches, then this would lead toward inconsistent global merged ontology. By exploiting disjoint knowledge in ontology O1, which restricts the concept Course as disjoint with Person, ontology mapping and merging systems should reject such initial mappings to avoid inconsistencies in merged ontology. Moreover, disjoint axioms together with equivalence axioms help validation of initial alignments and mappings found in first stages of ontology merging. For example, explicit descriptions about the disjointness of two concepts (C and D) in ontology O1 and equivalence of concepts (D' and F) in ontology O2 help to detect semantically inaccurate mapping between concepts (D and F).
5. Detecting Inaccurate Mappings by Instance matching strategy: Disjoint knowledge axioms help identifying the initial inaccurate instance based mappings that originate when same instance propose semantically distinct concept as merge candidate in the process of ontology merging. For example, consider ontologies O1 and O2 as shown in Fig. 5 where instance based matching technique identify concept Professor of ontology O1 as the candidate of merge with the concept Researcher of ontology concept O2 based on Identical Instance JOHN. These mappings could be rejected if merging system considers disjoint knowledge axioms in local ontologies that separate the domain of concepts Student and Staff (Faculty) while calculating similarities to produce mappings of semantically distinct concepts.

6. Better reasoning and inference mechanisms with disjoint-of axioms among properties: Due to the significance of disjoint-of axioms, W3C has included the construct disjoint-of to specify disjointness between properties and their hierarchies in new specie of OWL, i.e., OWL 2 that serves as current recommendation (2009) for building ontologies. During evaluation of current description logic reasoners, we observed that they do not fulfill the existing demands of enriched expressive ontologies with the constraints of disjointness and lack reasoning when used in real applications. Reasoning and inference with disjoint axioms between concepts and properties furnish more semantic power, spark the inference mechanism and provide better automatic reasoning capability to the ontology merging process, and help to build more well formed ontologies, which fulfill their purposes when used in applications. This requires that ontology merging systems should avoid all type of errors especially disjoint knowledge omission, common class or property in disjoint decomposition, redundancy of the disjoint-of relations among concepts and properties during construction of a merged global ontology from local ontologies.

Various versions of my system can be found at my personal site[7] 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.

5 Conclusion

This paper presents our analysis and role of disjoint knowledge axioms in the process of ontology merging. It explains and investigates our hypothesis that the ontology merging can be done better by considering disjoint partitions within the source ontologies. It provides a discussion on the different steps involved by DKP-AOM for the merging of heterogeneous ontologies. The whole merging process yields some significant points by the use of disjoint axioms in the overall process of merging. Disjoint knowledge axioms help to identify initial inaccurate mappings and remove ambiguity when concept with same symbolic identifier have different meaning in different local ontologies in the process of ontology merging. Disjoint axioms separate the knowledge in distinct chunks and enable concept matching within the boundaries of sub-hierarchies of the entire ontology concept hierarchy. While finding matches between concepts of ontologies, domain specific heuristics about the disjoint knowledge about concepts in source ontologies minimize the search space and thus reduce the runtime complexity of ontology merging. We conclude that disjoint knowledge analysis for the ontology merging is very much helpful for the detection of inconsistent initial mappings that originate from concept name or instance matching strategies. It reduces the search space for concept matching, and promotes the consistent computation by exploiting reliable logical inference on facts by axiomatization. We have also presented the results of DKP-AOM in OAEI 2015 as a success in the conference and OA4QA track ontologies by producing only consistent and coherent results.

[7] http://sites.google.com/site/mhdfahad

References

Anjum, N., Harding, J., Young, B., Case, K. (2010). Gap analysis of ontology mapping tools and techniques. In *Enterprise interoperability IV* (pp. 303–312). Berlin: Springer.

Baumeister, J., & Seipel, D. (2005). Smelly owls-design anomalies in ontologies. In *FLAIRS conference* (Vol. 215).

Brank, J., Grobelnik, M., Mladenic, D. (2005). A survey of ontology evaluation techniques. In *Proceedings of the conference on data mining and data warehouses (SiKDD 2005)* (pp. 166–170).

Chalupsky, H. (2000). Ontomorph: a translation system for symbolic knowledge. In *KR* (pp. 471–482).

El Jerroudi, Z., & Ziegler, J. (2008). imerge: interactive ontology merging. In *Proceedings of the international conference on knowledge engineering and knowledge management (EKAW 2008)* (p. 52).

Euzenat, J., Shvaiko, P., et al. (2007). *Ontology matching* (Vol. 333). Berlin: Springer.

Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C. (2011). Ontology alignment evaluation initiative: six years of experience. In *Journal on data semantics XV* (pp. 158–192). Berlin: Springer.

Fahad, M. (2015). Dkp-aom: results for oaei 2015. arXiv preprint arXiv:151001659.

Fahad, M. (2017). Merging of axiomatic definitions of concepts in the complex owl ontologies. *Artificial Intelligence Review*, *47*(2), 181–215. https://doi.org/10.1007/s10462-016-9479-5.

Fahad, M., & Qadir, M. A. (2008). A framework for ontology evaluation. *ICCS Supplement*, *354*, 149–158.

Fahad, M., Moalla, N., Bouras, A. (2011). Towards ensuring satisfiability of merged ontology. *Procedia CS*, *4*, 2216–2225.

Fahad, M., Moalla, N., Bouras, A. (2012). Detection and resolution of semantic inconsistency and redundancy in an automatic ontology merging system. *Journal of Intelligent Information Systems*, *39*(2), 535–557.

Jiménez-Ruiz, E., & Grau, B.C. (2011). Logmap: logic-based and scalable ontology matching. In *The Semantic Web–ISWC 2011* (pp. 273–288). Berlin: Springer.

Jiménez-Ruiz, E., Grau, B. C., Sattler, U., Schneider, T., Berlanga, R. (2008). *Safe and economic re-use of ontologies: a logic-based methodology and tool support*. Berlin: Springer.

Kim, J., Jang, M., Ha, Y.G., Sohn, J.C., Lee, S.J. (2005). Moa: owl ontology merging and alignment tool for the semantic web. In *Innovations in applied artificial intelligence* (pp. 722–731). Berlin: Springer.

Klein, M. (2001). Combining and relating ontologies: an analysis of problems and solutions. In *IJCAI-2001 Workshop on ontologies and info sharing* (pp. 53–62).

Kotis, K., Vouros, G. A., Stergiou, K. (2006). Towards automatic merging of domain ontologies: the hcone-merge approach. *Web Semantics: Science, Services and Agents on the World Wide Web*, *4*(1), 60–79.

Maiz, N., Fahad, M., Boussaid, O., Bentayeb, F. (2010). Automatic ontology merging by hierarchical clustering and inference mechanisms. In *Proceedings of I-KNOW* (pp. 1–3).

McGuinness, D.L., Fikes, R., Rice, J., Wilder, S. (2000). An environment for merging and testing large ontologies. In *KR* (pp. 483–493).

Mitra, P., & Wiederhold, G. (2002). Resolving terminological heterogeneity in ontologies. In *Proceedings of the ECAI workshop on ontologies and semantic interoperability*.

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

Qadir, M. A., & Noshairwan, W. (2007). Warnings for disjoint knowledge omission in ontologies. In *Second international conference on internet and web applications and services, 2007. ICIW'07* (pp. 45–45). IEEE.

Qadir, M. A., Fahad, M., Noshairwan, M. W. (2007). On conceptualization mismatches between ontologies. In *IEEE international conference on granular computing, 2007. GRC 2007* (pp. 275–275). IEEE.

Raunich, S., & Rahm, E. (2011). Atom: automatic target-driven ontology merging. In *2011 IEEE 27th international conference on data engineering (ICDE)* (pp. 1276–1279). IEEE.

Shvaiko, P., & Euzenat, J. (2013). Ontology matching: state of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, *25*(1), 158–176.

Stumme, G., & Maedche, A. (2001). Fca-merge: bottom-up merging of ontologies. In *IJCAI* (Vol. 1, pp. 225–230).

Völker, J., Vrandečić, D., Sure, Y., Hotho, A. (2007). Learning disjointness. In *The semantic web: research and applications* (pp. 175–189). Berlin: Springer.