Association in Level 2 fusion

Mieczyslaw M. Kokar[a], Christopher J. Matheus[b], Jerzy A. Letkowski[c], Kenneth Baclawski[d] and Paul Kogut[e]

[a]Northeastern University, ECE, Boston, MA, USA
[b]Versatile Information Systems, Inc., Framingham, MA, USA
[c]Western New England College, Springfield, MA, USA
[d]Northeastern University, CS, Boston, MA, USA
[e]Lockheed Martin, King of Prussia, PA, USA

ABSTRACT

After a number of years of intensive research on Level 1 fusion, the focus is shifting to higher levels. Level 2 fusion differs from Level 1 fusion in its emphasis on relations among objects rather than on the characteristics (position, velocity, type) of single objects. While the number of such characteristics grows linearly with the number of objects considered by an information fusion system, this cannot be said about the number of possible relations, which can grow exponentially. To alleviate the problems of computational complexity in Level 2 processing, the authors of this paper have suggested the use of ontologies. In this paper we analyze the issue of association in Level 2 fusion. In particular, we investigate ways in which the use of ontologies and annotations of situations in terms of the ontologies can be used for deciding which of the objects, and/or relations among such, can be considered to be the same. This is analogous to data association in Level 1 fusion. First, we show the kinds of reasoning that can be carried out on the annotations in order to identify various objects and possible coreferences. Second, we analyze how uncertainty information can be incorporated into the process. The reasoning aspect depends on the features of the ontology representation language used. We focus on OWL - the web ontology language. This language comprises, among others, constructs related to expressing multiplicity constraints as well as such features like "functional property" and "inverse functional property". We will show how these features can be used in resolving the identities of objects and relations. Moreover, we will show how a consistency-checking tool (ConsVISor) developed by the authors can be used in this process.

Keywords: Information fusion, Level 2, data association, ontology, inference

1. INTRODUCTION

One of the steps that is typically part of the processing in information fusion is the step called "data association".[1,2] The main idea of data association is that when two sources provide reports on a number of objects (also referred to as "individuals"), it is important to know when the reports refer to the same object in the world. This problem can occur at any level of processing,[3] from Level 0 through Level 4. For instance, when a sensor reports that there is an object O at location (x, y, z), the simple question is whether this location is the same as the one that another sensor refers to as (p, q, r). Another question is whether the ID of the object is the same as the one that this sensor refers to as P.

The issue of data association has been investigated for Level 1 fusion. In general, to determine the association between two data items, a measure of distance is computed and the association function is selected that minimizes the (weighted) distance over all possible associations.

While such an approach is feasible in Level 1 fusion, for higher-level fusion a different approach is required. The main reason is that in higher-level fusion quantitative features of particular individuals are often not available, since information about the individuals may be represented in text documents, which then are mapped into logical representations, rather than quantitative form. Consequently, the association process needs to be compatible

Further author information: (Send correspondence to M. M. Kokar): E-mail: kokar@coe.neu.edu, Telephone: 1 617 373 4849

with the form of representation, i.e., the process must be based on text analysis tools, like lexical analyzers and ontology-based annotators, and logical tools, like theorem provers, consistency checkers and others.

In this paper we discuss the issue of association for higher-level fusion. Although we consider text based documents as input, we don't go into any details of text processing tools. Instead, we consider only the logic based representations assuming that text has already been processed by an ontology-based annotator.

First we introduce a simple example of two text documents in which either direct or indirect references are made to some individuals. The text documents are then annotated using an ontology based tool, like AeroSWARM.[4] The annotation is based on an ontology. The ontology and the annotation are expressed in an ontology language called OWL (Web Ontology Language).[5] In the annotation, individuals are referred to by *individual names*.

The annotations of two documents can then be merged into one annotation. This step is analogous to the data fusion process of Level 1. It is very important to make sure that if references are made to the same individual in both documents, the merged annotation states the fact that this is the same object. It can be achieved either by using one name for the individual in the merged document, or by adding an assertion that the appropriate two individual names in the merged annotation refer to the same individual.

To resolve the question of whether two individual names denote the same object or not, we use the ConsVISor tool.[6] ConsVISor is a consistency checker of ontologies, i.e., it takes an ontology or an annotation as input and outputs that the input is either *consistent* or *inconsistent*. The scenario of checking the association of two individual names involves testing two hypotheses: (1) that the two names represent the same individual and (2) that the two names represent two different individuals. One of these two hypotheses is added to the annotation and ConsVISor is invoked. ConsVISor either finds the annotation consistent, which means lack of evidence against the added hypothesis, or inconsistent, which means existence of evidence against the added hypothesis.

The paper is organized as follows. First, in the next section we discuss ontologies and their use in information fusion (Section 2). Then in Section 3 we discuss an example of association in text based documents. In Section 4 we discuss constructs of the OWL language that can be utilized in reasoning about associations. In Section 5 we make some suggestions on how confidence of association can be quantified. And finally, in Section 6 we present our conclusions and directions for future research.

2. ONTOLOGIES AND LEVEL 2 FUSION

One of the issues in Level 2 fusion is the focus on relations among objects, rather than on the characteristics (position, velocity, type) of single objects. While the number of characteristics grows linearly with the number of objects considered by an information fusion system, this cannot be said about the number of possible relations, which can grow exponentially. To alleviate the problems of computational complexity in Level 2 processing, the authors of this paper have suggested the use of ontologies.[7–10] In this section we give a brief statement on what ontologies are and how they can be used in Level 2 fusion.

An ontology is an explicit, formal, machine-readable semantic model that defines the classes (or concepts) and their possible inter-relations specific to some specified domain.[11] To construct an ontology one must have an ontology specification language. The main driving force for using ontologies is the emergence of web-enabled agents.[11] These agents can reason about and dynamically integrate the appropriate knowledge and services at run-time based on formal ontologies. Ontologies are also the basis for the Semantic Web,[12] where they are being used to create machine-readable, semantic-descriptions of Web content that can be shared, combined and reasoned about automatically by theorem provers and intelligent agents.

As part of its Semantic Web effort, the W3C has developed a new XML-based language called the Web Ontology Language (OWL).[5] OWL is an emerging standard for ontologies and knowledge representations, based on the Resource Description Framework (RDF)[13] and the DARPA Agent Markup Language (DAML), which is the immediate predecessor of OWL. OWL is a declarative, formally defined language that fully supports specialization/generalization hierarchies as well as arbitrary many-to-many relationships. Both model theoretic and axiomatic semantics have been fully defined for the elements in OWL/DAML providing strong theoretical as well as practical benefits in terms of being able to precisely define what can and cannot be achieved with

these languages. The field is relatively young, yet several tools have been developed and many more are on the horizon for creating OWL ontologies and processing OWL documents. In our work, we create ontologies as UML diagrams and then programmatically convert them into DAML/OWL representations.[14]

Ontologies capture potential objects and potential relations; that is to say, they do not describe what is in *the world* but rather what can be in the world. Ontologies, however, can be used to annotate, or mark-up, descriptions of instances of the world in what are called instance annotations.

In Figure 1 we show an example of a very simple fragment of an ontology expressed in UML.[15] It does not include all the details which can be represented in an ontology representation language, like OWL.[5] While the OWL representation is appropriate for computer consumption, the UML representation, on the other hand, is more appropriate for humans. The aspects captured in this UML representation include classes, subclasses and associations. Classes are represented as boxes. Each class defines a "type", i.e., it describes objects that have some characteristics in common. In particular, it means that objects are linked to objects from other classes according to the *properties*, which in this diagram are shown as arrows (or just lines). A hollow arrow represents a special property called *subclass*.

In Figure 1 we see four classes and seven properties. The Leader class is a subclass of Person. This means every leader (object from the Leader class) is a person. The adornments at some of the ends of the lines represent the multiplicity constraints. For instance, this ontology shows that a leader can (actually must) lead exactly one organization. Moreover, every organization is led by exactly one leader. (Only the latter constraint is realistic, but this ontology is meant solely for illustration purposes and thus we did not try to accurately model the real world.) Moreover, since every leader is a person, every leader must have one name and one age.

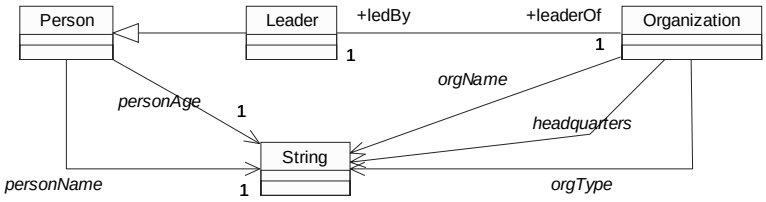

Figure 1. A Simple Ontology in UML

Below we show a small fragment of this ontology represented in OWL. The first part declares a class `Person`. The second part declares a property that this class has, `personName`, i.e., the name property.

```
<owl:Class rdf:ID="Person">
  <rdfs:label xml:lang="en">Person</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty rdf:resource="#personName"/>
```

```
     <owl:cardinality rdf:datatype=
    "http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty rdf:resource="#personAge"/>
     <owl:cardinality rdf:datatype=
    "http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="personName">
  <rdfs:label>person name</rdfs:label>
  <rdfs:comment>Name of a person.</rdfs:comment>
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range rdf:resource=
  "http://www.w3.org/2000/10/XMLSchema#string" />
</owl:ObjectProperty>
```

In the rest of this paper we make use of two ontologies - one ontology as shown in Figure 1 and a more relaxed version of this one in which the multiplicity constraints have been removed. In the next section we will see that such ontological commitments can make a difference in reasoning about associations.

3. EXAMPLES OF REASONING ABOUT ASSOCIATIONS

In this section we show a few examples of how ontologies can be used in the process of reasoning about associations, i.e., we will show how we can attempt to derive that either two individual names denote the same object or they denote two different objects in the world.

Now assume that we have two text documents in which references are made to two individuals whose role is "leader of an organization" and that the leaders are referred to by their IDs as `A1` and `A2`. The fact that the names associated with the leaders are `Osama` and `Strossen` may or may not be that relevant since it all depends on what world we live in. For instance, it depends on whether people are allowed to have multiple names, whether organizations can have multiple leaders and whether the same person can lead more than one organization.

Assume also that the text has been processed using an ontology based annotation tool, like AeroSWARM[4] and that the annotations generated by this tool have been merged into one annotation file. Below we show a small fragment of such a file.

```
<Leader rdf:ID="A1">
  <personName rdf:datatype=
  "http://www.w3.org/2000/10/XMLSchema#string">Osama</personName>
  <personAge rdf:datatype=
  "http://www.w3.org/2000/10/XMLSchema#string">old</personAge>
  <leaderOf rdf:resource="#X"/>
</Leader>
<Leader rdf:ID="B1">
  <personName rdf:datatype=
  "http://www.w3.org/2000/10/XMLSchema#string">Strossen</personName>
  <personAge rdf:datatype=
  "http://www.w3.org/2000/10/XMLSchema#string">mature</personAge>
  <leaderOf rdf:resource="#Y"/>
```

```
  <owl:sameAs rdf:resource="#A1"/>
</Leader>
```

This piece of annotation captures information about two individuals who are declared to be the leaders of organizations. The question is whether these two individual names (in this case we use the IDs `A1` and `B1`) denote two different individuals or just one? In other words, the hypothesis is that Osama and Strossen are two names of the same individual. To answer this question we inserted (see the line next to the last) a hypothesis that they are actually denoting the same individual. Then we tested this annotation using ConsVISor. As expected, ConsVISor derived that this annotation is inconsistent. ConsVISor's reasoning is based upon the knowledge that is represented in both the annotation and the ontology. The ontology in this case states that an organization can be led by exactly one leader and a leader must lead exactly one organization. This is not the case in the current annotation. But once the line `<owl:sameAs rdf:resource=‘‘#A1’’/>` of annotation is removed, ConsVISor declares that the annotation is consistent. In this paper we submit that ConsVISor could be used in the process of association in Level 2 fusion.

As our next step we analyzed possible scenarios of deciding whether two individual names denote the same individual or not. These possibilities can be explained using the decision tree shown in Figure 2. This tree represents various cases of the relation between an annotation and the ConsVISor decision with respect to the hypothesis about two (syntactic) representations (annotations). At the top of the Figure we have descriptions of three classification rules: *Ground Truth, Hypothesis, ConsVISor's Decision* and *Example*. The meaning of these rules is as follows:

Ground Truth This is the root node. It has two branches. The upper branch represents the situation in which two individual names describe the same individual. The lower branch represents the situation in which two individual names describe two different individuals. Note, however, that even when we know that we know this is the same individual, the annotation might be logically incomplete, i.e., it might not be possible to prove a theorem one way or the other. This kind of situation cannot be resolved by using tools of logic. So we need to limit our analysis to only two cases:

- Two names represent the same individual and the ontology has support for this.
- Two names represent two different individuals and the ontology has support for this.

Hypothesis: This is the statement about a hypothesis that is added to an annotation (a query). There are two possible hypotheses - either that the two names represent the same individual in the world, or that they represent two different individuals. This is expressed in OWL as `sameAs` and `differentFrom`.

ConsVISor's Decision: ConsVISor can decide that a given annotation (with the added hypothesis) is either "consistent" or "inconsistent".

Example: This represents a URL of an annotation for this example. Note, however, that if ConsVISor is correct, it should never conclude "inconsistent", when two names represent the same individual and there is support for such a conclusion in the ontology. So this branch of the decision tree is marked as "Impossible!". Similarly, it should not derive an inconsistency when there is support in the annotation for the conclusion that two names represent two different individuals. Consequently, the bottom branch of the tree is marked as "Impossible!".

The examples (the annotation files) listed in Figure 2 and then tested using ConsVISor take advantage of only one feature of OWL. The inconsistency derived by ConsVISor is based upon the fact that two annotations of the same individual must always declare the same properties and property values. In other words, if we have two annotations in which properties associated with one individual name are different and the `sameAs` clause is added to the annotation, ConsVISor will detect an inconsistency. Other features of OWL that can be used in the process of reasoning about association are discussed in Section 4.

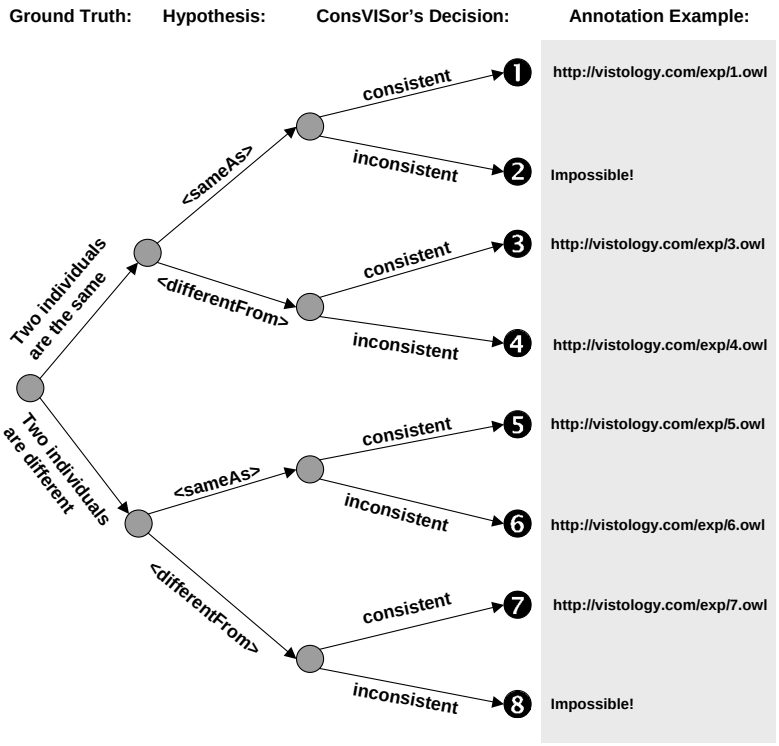

Figure 2. Possible Scenarios

4. ASSOCIATION THROUGH REASONING

The OWL language has a large number of constructs that can be used to specify the domain of interest. These constructs can be used to determine whether two names refer to the same individual. Being a formal language, OWL uses logical reasoning to make this determination. Generally speaking, when one considers whether two names refer to the same individual, there are three possible outcomes of the reasoning process:

1. The two names can be shown to refer to the same individual.
2. The two names can be shown to refer to different individuals.
3. It is consistent for either of the two cases to be true.

In this section we discuss the most important constructs of OWL that can affect this decision, and we give some examples to illustrate how this happens.

The OWL language specifies an individual name using a Uniform Resource Indicator (URI). A URI is a unique identifier. It uses the same syntax as URLs. URIs generalize URLs in the sense that they may not specify an Internet location, like URLs do. Characteristics of individuals (such as position or velocity) are specified using properties. Properties are themselves URIs. A property that specifies a characteristic of an individual is called a `DatatypeProperty`. One specifies a characteristic of an individual by two URIs and a value: the URI of the individual whose characteristic is being specified, the property that specifies the kind of characteristic, and the value of the characteristic.

Relations between individuals are also specified using properties. A property of this kind is called an `ObjectProperty`. A relation between two individuals is specified by three URIs: the URI of the property that specifies the kind of relation and the two URIs of the individuals that are being related.

There are two special OWL properties that can be used for explicit data association: `sameAs` and `differentFrom`. These explicitly specify that two URIs refer to either the same individual or different individuals. One can specify that a whole set of URIs refer to different individuals by using the `AllDifferent` construct.

OWL properties are always binary in the sense that they always relate one entity with another. For a datatype property, the two entities are the URI of an individual and the property value. For an object property, the two entities are the URIs of two individuals being related. The inverse of a property is the same as the property except that every pair of URIs is reversed. For example, the inverse of `leaderOf` is `ledBy`. Higher arity relations are synthesized by introducing a new URI to represent the instance of the relation. A series of binary relations are then used to relate this new entity to the set of individuals that are to be related with one another.

By default, every OWL property is many-to-many. A given individual can be related to any number of other values or individuals. Furthermore, unless one specifies constraints on an OWL property, it can relate any individual to any other value or individual. Without suitable constraints, no reasoner will be able to conclude that two individual names refer to either the same or different individuals, unless they were explicitly specified to be the same or different.

The two most important OWL constructs for data association are `FunctionalProperty` and `InverseFunctionalProperty`. Both of these are classes of properties. When one specifies that a property belongs to one of these classes, then that property is constrained to be either functional or inverse functional. If a property is functional then a given individual can be related to at most one other value or individual. In mathematical terms this means that the property is a partial function. If a property is inverse functional, then an individual or value can be used by at most one individual. In other words, the reverse (or inverse) property is functional. One specifies that a leadership is functional by asserting that `ledBy` is an instance of `FunctionalProperty` or that its inverse `leaderOf` is an instance of `InverseFunctionalProperty`. If one of these is specified, and if two individual names are both known to refer to the leader of the same organization, then those names must represent the same individual.

The next most important OWL construct for data association is the cardinality constraint. A cardinality constraint restricts the number of individuals (or values) that a given individual can be related to. A `maxCardinality` constraint specifies an upper bound on this number, a `minCardinality` constraint specifies a lower bound, and a `cardinality` constraint specifies the exact number. A `maxCardinality` constraint of 1 is essentially the same as a functional property constraint, except that a functional constraint is absolute while a cardinality constraint is only relative to the individuals in a particular class. The `maxCardinality` and `cardinality` constraints are especially useful for proving that two individual names refer to the same individual. They accomplish this in much the same way as the functionality constraints discussed above. The `minCardinality` constraint is not very useful for such a determination.

Another important OWL construct is the `disjointWith` property. When two classes are specified to be disjoint, then they cannot have any instances (individuals) in common. Disjointness conditions are useful for proving that two names must refer to different individuals. For example, if one specifies that Car and Airplane are disjoint, then every instance of Car is different from every instance of Airplane. An extreme form of disjointness is the `complementOf` construct. This specifies that two classes are not only disjoint, but also that every individual is in one of the two.

The remaining OWL constructs are less important for data association, but they can sometimes achieve such a result, especially when combined with other constructs. The `oneOf` construct is used for specifying that the members of a class must all come from a given list of instances. For example, if one knows all of the members of an particular organization, and if an individual is known to be a member of this organization, then one can conclude that the individual is the same individual as one of the known set of members. One can then reason by "process of elimination" to determine the precise member (i.e., if one can eliminate the other possibilities, then the remaining one must be the same as the individual). The `unionOf` construct specifies that a class is the union of some other classes. This is similar to the `oneOf` construct except that it uses classes instead of individuals.

The `unionOf` construct is much weaker than `oneOf`, but it is sometimes useful for data association via a similar process of elimination.

One can specify in OWL that one class is a subset of another using the `subClassOf` construct. If two classes are subsets of each other, then they have the same set of elements. This can be specified by either making two `subClassOf` assertions or by asserting that one class is an `equivalentClass` of the other. Similarly, a property can be a `subPropertyOf` another, and one property can be an `equivalentProperty` of another. None of these constructs can be used by itself for data association. However, they can be used to determine additional facts about an individual which could eventually lead to a data association. For example, if an individual is known to be an instance of a class that is a `subClassOf` another class which has a `oneOf` constraint, then one may be able to use a process of elimination to determine that the individual name refers to the same as another individual name.

The OWL `intersectionOf` construct is a strong form of the `subClassOf` constraint. For example, if one specifies that the AmphibiousVehicle class is the intersection of the LandVehicle and WaterVehicle classes, then one is not only specifying that AmphibiousVehicle is a subclass of the other two, but also that every instance of both of the other two is amphibious.

One can specify that a property has a `domain` and `range`. Thus if one observes that two individuals are related via a property, and if the property has domain and range constraints, then one can conclude that the first individual is in the domain and the second is in the range. The domain and range constraints are absolute in that they apply globally to the property. The OWL `allValuesFrom` constraint is a relative form of the range constraint.

One can specify that an OWL property is a `SymmetricProperty` or a `TransitiveProperty`. A symmetric property is equivalent to its inverse property. A transitive property is transitive as a binary relation. Both of these constraints allow one to infer new relationships from known relationships. When combined with functional or inverse functional constraints they can be used to prove that two individual names refer to the same individual.

If two classes are equivalent, it does not follow that they are the same, and similarly for properties. However, this could be evidence that they are the same. More generally, when one cannot logically determine whether two individual names refer to the same individual or to different individuals, one may be able to determine whether the two names are likely or unlikely to refer to the same individual. This kind of decision requires a different form of reasoning that is discussed in the next section.

5. CONFIDENCE

In this section we will further explore the results provided by ConsVISor and investigate how certain we can be of their meaning. In this first paragraph we assume complete confidence in the information contained in the annotations. Consider first the case where ConsVISor returns *consistent* for both hypotheses. Since it is consistent to believe either that they are the same or that they are different we cannot make any association claims at all; there simply is not enough information in the annotations to make any conclusions regarding their equivalence or non-equivalence. The converse case in which both hypotheses produce *inconsistent* results should never occur unless there is an inconsistency in the underlying ontology. The interesting cases occur when one hypothesis is consistent and the other is inconsistent. Take the case where the `sameAs` hypothesis is consistent and the `differentFrom` hypothesis is inconsistent. The first result does not really tell us much but, since the two references cannot be referring to different individuals (i.e. `differentFrom` is inconsistent) they must be coreferences. Conversely, if the `sameAs` hypothesis is inconsistent and the `differentFrom` hypothesis is consistent, the two references cannot refer to the same individual.

As we indicated, the previous paragraph assumed that all of the information in the annotations was known to be true. What happens in cases where the annotations may contain information of questionable certainty? This situation is more realistic when considering real world data sources and so we need to consider how we might approach this problem. We will assume in the examples that follow the use of a very simple confidence measure that can be associated with part of an annotation. For simplicity, and without lost of generality, we will use a certainty measure based on a scale from 0 to 1 with 0 representing no confidence and 1 representing total confidence. Let us now assume one of the documents contains annotations that are tagged with uncertainty

measures indicating some degree of confidence in the validity of the information. In the simplest case all the annotations may be given the same value representing the overall confidence in the source of the document. In other cases specific annotations may have their own uncertainty measures owing perhaps to the judgment made by the source of the information for example, an analyst notes a 90% confidence that the surveillance photo contains an A10 aircraft. The question now is, how do we make use of this uncertainty information in evaluating the results from ConsVISor?

There is no notion of uncertainty in OWL today and, as with all OWL reasoners, ConsVISor does not have any way to deal with uncertainty even if an ontology has an explicit way for incorporating uncertainty into its annotations. What ConsVISor can do, however, is provide us with the information about the annotations involved in the determination of an inconsistency. Using an external process it is possible in some cases to combine the uncertainties of the implicated annotations to make some statements about the uncertainty associated with an association claim. Consider the case where one document contains annotations that are known to be certain (i.e., certainty factor equal to 1 and the other document has annotations in which some are marked as having certainty measures of less than 1. Now take the case where both the `sameAs` and the `differentFrom` hypotheses produce *inconsistent* results (remember this was not possible above when everything was certain but it can occur when there is uncertainty). If it is the case that the annotations involved in one of the results are all known for certain whereas some of the annotations in the other result are not, we can conclude that the first results are correct and use them to make the appropriate association claim. If it is the case that both results contain uncertain annotations one could use a heuristic that picks the result that is dependent on the least uncertain annotations and offer it as a "more probable" coreference result. For example, if the `sameAs` hypothesis produced an *inconsistent* result based on annotations whose combined certainty factors amounted to less than the combined certainties of the `differentFrom` hypothesis, a suggestion could be made that the two references are "more likely" to be coreferences than not. Note that this heuristic has not been formally analyzed or tested and is simply suggested here as a possible first approach from which to base future research.

The other cases are perhaps less interesting. If you have uncertainty when both results turn up *consistent* you cannot say anything more or less than you could under certain conditions. If you have uncertainty involved in the annotations implicated in an *inconsistent* result when the other result is *consistent*, you now no longer can state with certainty that the references are equivalent or not. It would be nice to return a measure of the likelihood of an association, but the determination of a sound method for calculating a reliable measure for the uncertainty of an association conclusion based on implicated annotations containing uncertainty is an open issue for future research.

6. CONCLUSION

In this paper we demonstrated a technique for discovering associations (coreferences) in documents marked up in OWL. The method assumes the existence of an ontology that appropriately defines the constraints that exist among the individuals in the domain of interest. Because OWL is a formal language tools can be developed that can reason about the contents of OWL documents, which is what our consistency-checking tool, ConsVISor, does. With some simple examples we demonstrated how we could use a tool like ConsVISor to test hypotheses about whether or not two individual names in different documents can be references to the same individual. This is achieved by combining the documents into a single document and adding an annotation that equates the individual names (using `sameAs`) or makes them different (using `differetFrom`). When this combined annotation is processed by ConsVISor and it reports an inconsistency we can conclude that the added annotation must be false; for example if the *inconsistent* result arose after adding the hypothesis that the individual names refer to different individuals then we can conclude that they in fact refer to the same individual.

Our examples used a single OWL construct, the cardinality constraint, to detect inconsistencies. We discussed ways in which other constructs may be used in a similar manner and we intend to explore such opportunities in future experiments. It is also possible to extend our approach to situations in which uncertainty information is included in the annotations. We discussed some thoughts on how this might be done in the absence of a formal way of representing uncertainty in OWL. In future work we intend to experiment with various ways of combining uncertainty metrics to provide a measure of confidence to the association conclusions that are made. This paper focused on the process of comparing just two individual names for coreference. A working system employing this

technique would need to apply this process to all pairs of individual names that could possibly be coreferences; this would include any pair of individuals that share a common class at some point in the class hierarchy captured in an ontology. Since everything in OWL is a member of the class Thing, one could attempt to coreference all pairs, but this would be computationally intractable. We will therefore need some way of limiting the search to some set of appropriate classes; this is yet another area to be addressed in future work.

Acknowledgments

Some of this material is based upon work supported by the Air Force Research Laboratory, Contract Number F30602-00-C-0188. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

REFERENCES

1. D. L. Hall, *Mathematical Techniques in Multisensor Data Fusion*, Artech House, Boston - London, 1992.
2. E. Waltz and J. Llinas, *Multisensor Data Fusion*, Artech House, Norwood, MA, 1990.
3. A. N. Steinberg and C. L. Bowman, "Revision to the JDL data fusion model," in *Handbook of Multisensor Data Fusion*, D. L. Hall and J. Llinas, eds., pp. 2–1–2–19, (CRC Press), 2001.
4. "AeroSWARM." Available at `http://ubot.lockheedmartin.com`.
5. OWL-ref, "OWL Web Ontology Language Reference," 2003. `www.w3.org/TR/owl-ref`.
6. K. Baclawski, J. A. Letkowski, C. J. Matheus, and M. M. Kokar, "The ConsVISor consistency checking tool." Available at `vistology.com/consvisor.html`.
7. C. J. Matheus, M. M. Kokar, and K. Baclawski, "A core ontology for situation awareness," in *Proceedings of the Sixth International Conference on Information Fusion*, pp. 545 –552, 2003.
8. C. J. Matheus, K. P. Baclawski, and M. M. Kokar, "Derivation of ontological relations using formal methods in a situation awareness scenario," in *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2003*, pp. 298–309, SPIE, 2003.
9. M. M. Kokar and J. Wang, "Using ontologies for recognition: An example," in *Proceedings of the Fifth International Conference on Information Fusion*, pp. 1324 – 1343, 2002.
10. M. M. Kokar and J. Wang, "An example of using ontologies and symbolic information in automatic target recognition," in *Sensor Fusion: Architectures, Algorithms, and Applications VI*, pp. 40–50, SPIE, 2002.
11. D. McGuinness, "Ontologies and online commerce," *IEEE Intelligent Systems* **16**(1), pp. 8–14, 2001.
12. T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American* , May 2001.
13. RDF, "Resource description framework (RDF) model and syntax specification," Feburary 1999. `www.w3.org/TR/REC-rdf-syntax`.
14. P. Kogut, S. Cranefield, L. Hart, M. Dutra, K. Baclawski, M. Kokar, and J. Smith, "UML for ontology development," *The Knowledge Engineering Review* **17, 1**, pp. 61–64, 2002.
15. "OMG Unified Modeling Language specification, version 1.3," tech. rep., Object Management Group, 1999.