Ontology Integration with Contextual Information

Dan Wu
Software and Computer Systems, School of Information and Communication Technology, Stockholm, Sweden

Keywords: Ontology Integration and Merging, Non-contradictory Ontology Integration, Context, Rules.

Abstract: By studying ontologies in an ontology repository, context rules are developed to improve ontology integration result. A context rule contains conditions for identifying a context. These context conditions are described by, so called, context criteria, which are, e.g., author and domain of an ontology. When the conditions, in a rule, are met, the rule is fired and the contextual information, in the body of the rule, is inserted into the reasoner, which is used for ontology integration. An example shows the construction of a context rule. The rule is used for an ontology integration. The integration result is indeed improved comparing with integration without contextual information.

1 INTRODUCTION

Semantic ontology integration and matching is important in many applications such as the semantic web and the knowledge and information integration (Euzenat et al., 2007). Although methods (Meilicke et al., 2007; Aleksovski et al., 2006 a,b) have been proposed, improvements are needed.

An approach of using contextual information is proposed here to improve the ontology integration. Our previous works show that different ontologies are considered as different terminology definitions from various perspectives (Håkansson and Wu, 2013); perspectives are then described with context and used for reasoning among different ontologies in a repostitory (Wu, 2013).

The paper is structured in five parts. Part 2 discusses related work. In part 3, the repository and context in the repository is presented. Part 4 shows an example of building contextual rules and how to apply them to integrate ontologies; in part 5, the discussion is given, and the last part concludes the work and present future work.

2 RELATED WORK

Meilicke and Stuckenschmidt (2007) combined numerical opimization and logical reasoning to improve the ontology matching result. It uses hungarian method to compute the optimal one-to-one mapping and then check if the mapping is consistent with the union of the ontologies. The idea of consistency checking is very similar to our apporach. However, we use rules to resolve the inconsistency, the results are different. And we create an integrated ontology that extend the previous ontologies, while Meilicke and Stuckenschmidt generate a mapping between two ontologies.

Aleksovski (2006 a, b) used the background knowledge for ontology matching. The background knowledge is an ontology covering the source and target ontologies and with rich semantic conceptual descriptions of the domian. In our approach, the contextual information is an ontology, however identified by context criteria, which domain is only one criterion. Context can be identified by a combination of seven criteria. The reasoning processes in two approaches are also different.

In an industrial scenario, a context ontology is used to improve the matching result of an enterprise ontology and a target ontology (Lin el al., 2010). Abstract context and operational context are modelled. During the matching process, the enterprise ontology, the target ontology and the context ontology are matched in turn. The overlapping terminologies of these ontologies are assigned with different weights or altering algorithms. This work uses a context ontology, however, created for the specific scenario. It is not certain if the context ontology can be reused or not.

Wu D..
Ontology Integration with Contextual Information.
DOI: 10.5220/0004820504850490
In *Proceedings of the 6th International Conference on Agents and Artificial Intelligence* (ICAART-2014), pages 485-490
ISBN: 978-989-758-015-4

The processes and the algorithms of handling the match and using context are also different.

3 CONTEXT AND REASONING

An OWL 2 ontology repository is used for reusing and integrating ontologies (Wu, 2013). The ontologies in the repository can be integrated to generate ontology intersections that covering broader and deeper descriptions of ontologies (*ibid.*)

3.1 Context and the Context Rules

The ontology and ontology intersections in the repository can be used as context for other ontology integration. Context criteria describe the context. Instead of using a term, criteria are used, such as, components of ontologies, metadata and administrative data of ontologies. The components of an ontology are entities of ontologies, e.g., a class or a property. The metadata of ontologies are domain information, author of ontology, purpose of ontology, context agent and time for creating the ontology. Context agent is the context user who can be either a person or an application. The administrative data is the ontology and the context identification. In the repository, the context criteria are represented as *entity of ontology*, *domain of ontology*, *author of ontology*, *purpose of ontology*, *context agent*, *time of ontology*, *OntologyIdentification and Contextidentification*. The context criteria may be used all together or with any combinations of the criteria. Thus, context is defined as:

Context := {ontologyEntity} | ontologyDomain | ontologyAuthor | ontologyPurpose | contextAgent | time | ontologyIdentification | contextIdentification

Context rules relate the situation of the ontology integration with ontologies in the repository. The rules are in the form of *if context () then contextualInformation*. The function *context ()* uses context criteria to delimit situations. The *contextualInformation* is a collection of expressions and axioms from an ontology or an ontology intersection.

For example, if several ontologies define class *"author"* and *"paper"* for *conference* domain, the contextual information extracted from the ontologies or ontology intersections is presented in a context rule as:

If context (entities ("author", "paper") and domain ("conference"))

Then contextualInformation (Author ⊏ ConferenceMember; Author ⊏User; submitPaper (Author,Paper); writePaper ≡ hasAuthor¯ ; hasAuthor(Paper,Author);
contributes (Person,Conference_document); has_authors ≡ contributes¯ ; has_authors(Conference_document, Person); etc.…)

This rule fires when a situation that is described with "*author*" and "*paper*" entities and the domain "*conference*". The contextual information is, then, used for the reasoning for integrating ontologies.

Context rules are used to build up the context knowledge. Context knowledge tackles the conceptual and the pragmatic heterogeneities of ontologies. For example, different ontology authors create different ontologies for the same domain. Then, the different perspectives of the authors are integrated and defined in the consequence part of the context rule. The integrated ontologies cover a broader and deeper conceptual description of the initial ontologies (Wu, 2013). These integrated ontologies are, then, used in context rules.

3.2 The Reasoning Process

The process for integrating task of two ontologies is described below, see Figure 1.

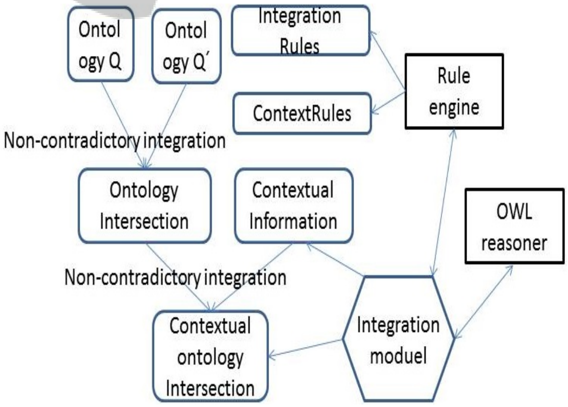

Figure 1: Process of contextual ontology integration.

From left to right, the integration starts with two input ontologies Q and Q'. The ontology intersection is produced through the non-contradictory process (Wu, 2013). The generated ontology intersection is an independent ontology with its own identification in the repository (*ibid.*). Then context rule is identified to refine the ontology intersection result.

By examining the ontology intersection from the first step, the knowledge expert chooses the proper context criteria values. Through the integration module, the values are added into the rule engine.

The rule engine fires the most specific rule, i.e., the most conditions are satisfied by the context criteria. The integration module will take the *contextualinformation* from the fired rule, and add it as the contextual information into the OWL reasoner to help to infer and refine the integration of the ontology intersection. This refining process is an integration of two ontologies, between the ontology intersection and the contextual information, with the help of integration rules. Finally, the integration module generates the result, the contextual ontology intersection. The process starts with two ontologies and ends with the contextual ontology intersection as the result.

4 AN EXAMPLE

This section describes an example to illustrate the process of the contextual ontology integration. The task is to integrate two ontologies using a context rule. Ontologies are downloaded from OAEI 2012 ontology evaluation website (http://oaei.ontologymatching.org/2012/conference/index.html). Three ontologies, “cmt.owl”, “cocus.owl” and “conference.owl”, are downloaded first in the repository and used for generating a context rule. New ontologies are downloaded later for showing the integration with the context rule.

4.1 The Context Rules

By integrating the three ontologies with the non-contradiction process, an intersection is generated. Since all ontologies describe conference domain, a context rule is generated for the domain of conference. The ontologies are integrated two by two because the non-contradiction process takes only two ontologies at a time. In other words, “cmt.owl” is integrated with “cocus.owl” first. The intersection of them is then integrated with the third ontology “conference.owl”. The result is an intersection of the three ontologies. Since terms of "Author" and "Paper" are commonly used in the domain of conference, this example uses only the definitions concerning these two terms. All the descriptions that contain "Author" and "Paper" are extracted from the ontologies. Protégé is used to extract the class usages and the result is presented in Table 1.

Table 1: The ontologies in the repository.

Cmt.owl	*Author* ⊏ ConferenceMember *Author* ⊏ User AuthorNotReviewer ⊏ *Author* Co-author ⊏ *Author* ∀x,y: (x,y) ⊙ (submitPaper)^{op} implies x ⊙*Author* and y ⊙*Paper* (*writePaper*)^{op}={(x,y)|(y,x) ⊙ (*hasAuthor*)^{op}} ∀x,y: (x,y) ⊙ (hasAuthor)^{op} implies x ⊙ *Paper* and y ⊙ *Author* ∀x,y: (x,y) ⊙ (markConflictOfInterest)^{op} implies (x ⊙ *Author* | x ⊙ *Chairman* | x ⊙ *Reviewer)* and y ⊙ *Paper* *Paper* ⊏ Document *Paper*⊓Review=∅ PaperAbstract ⊏ *Paper* PaperFullVersion ⊏ *Paper* ∀x,y: (x,y) ⊙ (acceptPaper)^{op} implies x ⊙ *Adminstrator* and y ⊙ *Paper* ∀x,y: (x,y) ⊙ (acceptedBy)^{op} implies x ⊙ *Paper* and y ⊙ Administrator *acceptPaper ≡acceptedBy*$^{-}$ ∀x,y: (x,y) ⊙ (assignedTo)^{op} implies x ⊙ *Paper* and y ⊙ Reviewer ∀x,y: (x,y) ⊙ (hasBeenAssigned)^{op} implies x ⊙ *Reviewer* and y ⊙ *Paper* ∀x,y: (x,y) ⊙ (co-writePaper)^{op} implies x ⊙ Co-author and y ⊙ *Paper* ∀x,y: (x,y) ⊙ (hasBid)^{op} implies x ⊙ *Paper* and y ⊙ Bid ∀x,y: (x,y) ⊙ (hasCo-author)^{op} implies x ⊙ *Paper* and y ⊙ Co-author ∀x,y: (x,y) ⊙ (hasDecision)^{op} implies x ⊙ *Paper* and y ⊙ Decision ∀x,y: (x,y) ⊙ (readByReviewer)^{op} implies x ⊙ *Paper* and y ⊙ Reviewer ∀x,y: (x,y) ⊙ (readPaper)^{op} implies x ⊙ *Reviewer* and y ⊙ *Paper* ∀x,y: (x,y) ⊙ (rejectPaper)^{op} implies x ⊙ *Administrator* and y ⊙ *Paper* ∀x,y: (x,y) ⊙ (rejectBy)^{op} implies x ⊙ *Paper* and y ⊙ Administrator
Cocus.owl	*Author* ⊏ User Corresponding_Author ⊏ *Author* *Paper* ⊏ Document Abstract ⊏ *Paper* Full_Paper ⊏ *Paper* Invited_Paper ⊏ *Paper* Short_Paper ⊏ *Paper*
Conference.owl	*Regular_author*⊏Conference_contributor Contribution_1th-author ⊏ *Regular_author* Contribution_co-author ⊏ *Regular_author* ∀x,y: (x,y) ⊙ (*has_authors*)^{op} implies x ⊙ Conference_Document and y ⊙ Person ∀x,y: (x,y) ⊙ (*contributes*)^{op} implies x ⊙ Person and y ⊙ Conference_document *Paper*⊓Extended_abstract = ∅ *Paper* ⊏ Regular_contribution ∀x,y: (x,y) ⊙ (*has_an_abstract*)^{op} implies x ⊙ *Paper* and y ⊙ Abstract ∀x,y: (x,y) ⊙ (is_the_1th_part_of)^{op} implies x ⊙ Abstract and (y ⊙ *Paper* | y ⊙ *Poster* | y ⊙ *Presentation)*

First, the signatures of ontology "cmt.owl" and "cocus.owl" are extracted and four sets of entities are produced: two sets are their labels of the classes; and two sets are their labels of object properties. Second, the comparison of syntax of the sets of labels of the classes and the object properties are conducted. Synonyms are identified from WordNet (http://wordnetweb.princeton.edu/perl/webwn). The string-identical labels and synonyms from the entity candidates are used for generating assumptions. The third step is to test the assumptions with the ontologies one by one and check the contradiction between them. Integration rules are used to handle the process and conflicts. An integration rule is for example: ***If*** *string-identical (Q.c, Q'.c') and synonyms (Q.m. Q'n) and (Q.c ⊏ Q.m | Q.m ⊏ Q.c) and (Q'.c' ⊏ Q' n | Q'.n ⊏ Q'.c')* ***then*** *c ≡ c'*. This rule says that to merge two string identical class c and c' in two ontologies if they share either the super classes that are synonyms or children classes that are synonyms. The non-contradiction part is preserved in the ontology intersection of ontologies "cmt.owl" and "cocus.owl". The ontology intersection is then integrated with "conference.owl" following the same non-contradiction process to generate the final intersection of the three ontologies. The ontology intersection of the three ontologies is used for building a context rule.

The rule condition is domain ("conference") and entities ("Author", "Paper"). The consequence part of the rule is extracted from the ontology intersection of the three ontologies. The rule is shown in Table 2.

Table 2: The contextual rule.

If context (domain("conference") and entity ("Author", "Paper")) ***Then*** *contextualInformation(* **∀x,y: (x,y) ⊙ (hasAuthor)^{op} implies x ⊙ *Paper* and y ⊙ *Author*; ∀x,y: (x,y) ⊙ (writePaper)^{op} implies x ⊙ *Author* and y ⊙ *Paper*; *writePaper*≡*hasAuthor*¯**;∀x,y: (x,y) ⊙ (submitPaper)^{op} implies x ⊙*Author* and y ⊙*Paper*; ∀x,y: (x,y) ⊙ (*has_authors*)^{op} implies x ⊙ Conference_Document and y ⊙ Person; ∀x,y: (x,y) ⊙ (*contributes*)^{op} implies x ⊙ Person and y ⊙ Conference_document; *has_authors* ≡ *contributes* ¯ ; **Author⊏Person**; Paper⊏Regular_contribution; Regular_contribution⊏Written_contribution; Written_contribution⊏Conference_contribution; Conference_contribution ⊏Conference_document; etc…..)

In the consequent part of the rule, only part of the intersection is shown because of the limit of space. The expressions, in bold font, are used for contextual integration carried out later on.

4.2 Integration with the Context Rule

Ontologies *confious.owl* and *confof.owl* are then downloaded and the task is to integrate them using the context rule shown in Table 2. Only classes and properties that contain terms "Author" and "Paper" are extracted from the ontologies and used in the example, shown in Table 3. They represent ontologies Q and Q' in section 3.2. In order to generate a contextual ontology intersection of these two ontologies, an intersection according to the non-contradictory integration process (Wu, 2013) is produced first. The result is shown in Table 4.

Table 3: Ontologies for showing contextual integration.

Confious.owl	∀x,y: (x,y) ⊙ (*has_author*)^{op} implies x ⊙ Article and y ⊙ Human;
	∀x,y: (x,y) ⊙ (*is_author_of*)^{op} implies x ⊙ Human and y ⊙ Article;
	has_author ≡ *is_author_of*¯ ; *Paper* ⊏ Article;
	abstract_of_*paper* ⊏ Article;
	Accepted_*Paper* ⊏ *Paper*
	Rejected_*Paper* ⊏ *Paper*
	Undecided_*Paper* ⊏ *Paper*
	∀x,y: (x,y) ⊙ (*is_concerned*)^{op} implies y ⊙ *Paper*; *Contact_Person* ⊏ Human

Confof.owl	∀x,y: (x,y) ⊙ (*writes*)^{op} implies x ⊙ *Author* and y ⊙ Contribution; ∀x,y: (x,y) ⊙ (*writenby*)^{op} implies x ⊙ Contribution and y ⊙ *Author*; *writes* ≡ *writtenby*¯; *Author* ⊏ Person; *Paper* ⊏ Contribution; *Paper*⊓Poster=∅; *Paper*⊓short_*paper*=∅

If only one definition exists in either ontology, the definition is per se true and preserved in the intersection. The string identical entity, e.g., paper, is integrated as one class in the intersection. The integration does not contradict the input ontologies. That is to say, classes "Accepted_Paper", "Rejected_Paper" and "Undecided_Paper" are subclasses of class "Contribution", which cause non contradiction to both original ontologies. The ontology intersection, in Table 4, is then been examined for identifying the context rule.

Table 4: Ontology intersection of Confious.owl & Confof.owl.

Paper ⊏ **Article**; *Paper* ⊏ Contribution; Accepted_*Paper* ⊏ *Paper*; abstract_of_*paper* ⊏ Article; Rejected_*Paper* ⊏ *Paper*; Undecided_*Paper* ⊏ *Paper; Paper*⊓Poster=∅; *Paper*⊓short_*paper*=∅; *Author* ⊏ Person ;
Contact_Person **⊏ Human;**
∀x,y: (x,y) ⊙ (*writes*)^{op} implies x ⊙ *Author* and y ⊙ Contribution;
∀x,y: (x,y) ⊙ (*writenby*)^{op} implies x ⊙ Contribution and y ⊙ *Author*;
***writes≡writtenby*$^{-}$**;
∀x,y: (x,y) ⊙ (*has_author*)^{op} implies x ⊙ Article and y ⊙ Human;
∀x,y: (x,y) ⊙ (*is_author_of*)^{op} implies x ⊙ Human and y ⊙ Article;
***has_author≡is_author_of*$^{-}$;**
∀x,y: (x,y) ⊙ (*is_concerned*)^{op} implies y ⊙ *Paper*;

When context condition data, i.e., domain is "conference" and entities are "paper" and "author", are entered into the rule engine, the context rule in Table 2 fires. The contextual information in the rule is then inserted into the OWL reasoner to help further integrating the ontology intersection. This process is actually an integration of the ontology intersection with the contextual information. The process of non-contradiction is applied also for this integration.

With OWL 2, inverse object properties can be defined, such as “has_author” and “is_author_of” are inverse properties. That is to say, if two individuals x and y are related by “has_author”, then y and x are related by “is_author_of”. One integration rule about the inverse relation is:

ObjectProperty-Inverse-rule:
If *O: r1 (A, B) and O: r2 (B, A) and inverseProperties (O: r1, O: r2) and O: r3 (B, A) and O: r4 (A, B) and inverseProperties (O: r3, r4)*
AND
Q: r1 (A, B) and Q: r3 (B, A) and inverseProperties (Q:r1, Q: r3)
Then *r2(B, A) and r4 (A, B) and inverseProperties (r2, r4)*

The rule is shown in functional-style syntax, rather than using DL syntax as in the tables. O and Q symbolize two general ontologies. For example, O: r1 (A, B) states that the class A is the domain of the property r1 in ontology O, and the class B is the range of the property r1 in O. InverseProperties (O: r1, O: r2) means that r1 and r2 are inverse properties. This integration rule shows the transfer of inverse role relations in two ontologies. A and B are two string identical classes in ontology O. Properties r1, r2, r3 and r4 are four object properties. If there are two definitions of inverse properties in one ontology as *inverseProperties(O:r1. O:r2)* and *inserseProperties(O:r3, O:r4)*; and if in the other ontology, the inverse property is between the inverse properties from the previous ontology, i.e., *inversProperties(Q:r1, Q:r3)*; and all definitions are about the string-identical two classes A and B. The inverse property will transfer to the other unrelated properties, i.e., *invserProperties(r2, r4)*. In the rule condition, namespaces O and Q are used. In the result part, the namespace is escaped since the result is the new independent ontology, the ontology intersection.

Back to the example, there two pairs of inverse properties are identified in Table 4: *InverseProperties (writes, writtenby)* and *InverseProperties (is_author_of, has_author).* From the contextual information in Table 2, *InverseProperties (writePaper, hasAuthor)* is identified. "*hasAuthor*" is string-identical to "*has_author*". However, the rule conditions are not completely fulfilled. The property "*writes"* and "*writePaper"* are not equivalent. The integration module asks the knowledge expert in such a situation: *EquivalentProperties(writes,writePaper)?*. When the knowledge expert confirms the axiom, it is added into the ontology intersection.

Another axiom, which needs to be confirmed, is *EquivalentClasses(Contact_Person, Author)?*. The class "*Contact_Person*" is defined in confious.owl, where a contact_person is a human and is_author of an article. And it is not defined in other ontologies; therefore, it is not a problem to add this axiom. When this axiom is confirmed, the integration rule is fired.

Accordingly, the new knowledge in the result is: "*is_author_of*" is the inverse property "*writtenby*", shown below:

∀x,y: (x,y) ⊙ (writterBy)^{op} implies x ⊙ Paper and y ⊙ Author;
∀x,y: (x,y) ⊙ (is_author_of)^{op} implies x ⊙ Contact_Person and y ⊙ Paper;
is_author_of≡writtenBy^{-};
Author⊏Human; writes ≡ writePaper; Contact_Person ≡Author

The final result is a non-contradictory ontology intersection under the open world assumption, shown below:

Article≡Contribution; abstract_of_paper⊏ Article; Rejected_Paper ⊏ Paper;

Undecided_Paper ⊏ Paper; Accepted_Paper ⊏ Paper; Paper ⊓ Poster = ∅; Paper ⊓ short_paper = ∅; Paper ⊏ Regular_contribution; Regular_contribution ⊏ Written_contribution; Written_contribution ⊏ Conference_contribution; Conference_contribution ⊏ Conference_document; Author ⊏ Person; Contact_Person ⊏ Human; Contact_Person ≡ Author; ∀x,y: (x,y) ⊙ (contributes)^{op} *implies x ⊙ Person and y ⊙ Conference_document; Contribute ⊓ has_author=∅; ∀x,y: (x,y) ⊙ (has_author)*^{op} *implies x ⊙ Article and y ⊙ Human; ∀x,y: (x,y) ⊙ (is_author_of)*^{op} *implies x ⊙ Human and y ⊙ Article; ∀x,y: (x,y) ⊙ (is_author_of)*^{op} *implies x ⊙ Contact_Person and y ⊙ Paper;is_author_of ≡ writePaper≡writes; ∀x,y: (x,y) ⊙ (is_concerned)*^{op} *implies y ⊙ Paper; ∀x,y: (x,y) ⊙ (submitPaper)*^{op} *implies x ⊙Author and y ⊙Paper; ∀x,y: (x,y) ⊙ (writePaper)*^{op} *implies x ⊙ Author and y ⊙ Paper; ∀x,y: (x,y) ⊙ (writes)*^{op} *implies x ⊙ Author and y ⊙ Contribution; ∀x,y: (x,y) ⊙ (writenby)*^{op} *implies x ⊙ Contribution and y ⊙ Author; ∀x,y: (x,y) ⊙ (writterBy)*^{op} *implies x ⊙ Paper and y ⊙ Author; writes≡writtenby ¯ ; has_author≡is_author_of ¯ ;has_author≡writes¯; has_author≡writePaper¯; writePaper ≡ writtenBy ¯ ; is_author_of≡writtenBy ¯ ; is_author_of ≡ writtenBy¯;*

With the help of the contextual information, the ontology integration result has been improved because of more available information. The contextual ontology integration result is covering even more axioms than the first time ontology integration result. As we can see, the result contains non-contradictory axioms from the first intersection of the two ontologies and the contextual information. The result contains as well new axioms from the integration rules, which do not contradict the original ontologies.

Some approaches create ontologies to describe context. We use context criteria. Using criteria for context is better than using a single term for describing context. The context can be described by the different combination of the criteria. In this sense, the context is dynamic identified. To create a context ontology costs more comparing with reusing ontologies and ontology intersection in a repository. And it is very uncertain if the context ontology can be reused in future or not. Building the context condition and the contextual information into rules for reusing is an improvement for using context in reasoning.

5 CONCLUSIONS AND FUTURE WORK

This paper shows how contextual information can be used for improving ontology integration using context rules. The result is promising: the example shows that broader and deeper conceptual definitions are deduced from the initial ontologies and the implied contextual information become explicit in the final result.

However, more tests need to be conducted to validate the context rules and the integration rules. Another important work is to develop methods to automatically discern the context and suggest the context criteria, which is dependent on the knowledge expert at the current stage.

REFERENCES

Aleksovski, Z., Klein, M., ten Kate, W., and van Harmelen, F., 2006. *Matching unstructured vocabularies using a background ontology.* s.l., Knowledge Engineering and Knowledge Management (EKAW).

Aleksovski, Z., ten Kate, W., and van Harmelen, F., 2006. *Expliting the structure of background knowledge used in ontology matching.* s.l., International Workshop on Ontology Matching (OM-2006).

Euzenat, J., SHvaiko, P., 2007. *Ontology Matching.* Berlin, Heidelberg: Springer-Verlag.

Håkansson, Anne; Wu, Dan, 2013. Conceptual Ontology Intersection for Mapping and Alignment of Ontologies. In: *Agent and Multi-Agent Systems in Distributed Systems - Digital Economy and E-Commerce.* s.l.:Springer Berlin Heidelberg, pp. 105-124.

Lin, Feiyu, Butters, Jonathan, Sandkuhl, Kurt, Ciravegna, Fabio, 2010. *Context-based Ontology Matching: Concept and Application Cases.* s.l., 2010 10th IEEE International Conference on Computer and Information Technology (CIT 2010).

Meilicke, C. and Stuckenschmidt, H., 2007. *Applying logical constraints to ontology matching.* s.l., KI2007: Advances in Artificial Intelligence.

Wu, D., 2013. *Ontology Integration with Non-Violation Check and Context Extraction,* Stockholm: KTH Royal Institute of Technology.