GMRs: Reconciliation of Generic Merge Requirements in Ontology Integration

Samira Babalou[0000−0002−4203−1329], Birgitta König-Ries[0000−0002−2382−9722]

Heinz-Nixdorf Chair for Distributed Information Systems
Institute for Computer Science, Friedrich Schiller University Jena, Germany
{samira.babalou,birgitta.koenig-ries}@uni-jena.de

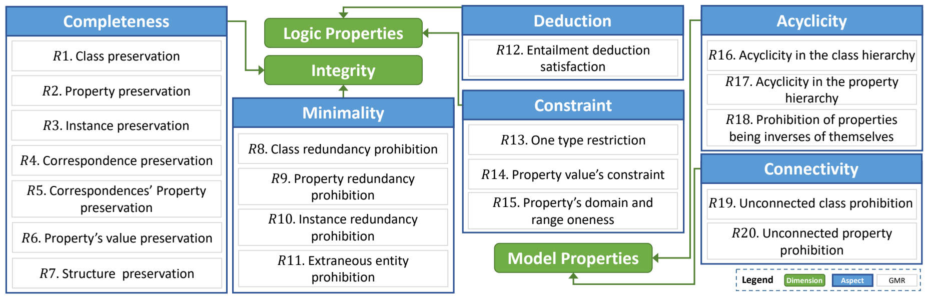
Abstract. With growing popularity of ontologies as semantics-aware integration solutions, various ontology merging methods have been proposed. Each of them uses their own set of criteria to determine a desirable merge result. In this work, we first categorize these criteria into generic merge requirements (GMRs). Not all of these requirements can be met simultaneously. We argue that users should be able to select those requirements that are most important to them, but that system support is required to determine a compatible set of requirements based on user inputs. We propose a graph-theory based approach to determining such sets.

Keywords: Ontology merging . Merge requirements . Graph theory

1 Introduction

An ontology is a formal explicit description of a domain. It contains a set of *entities* including classes, properties, and instances. Given a set of input ontologies and a set of correspondences between them, an ontology merging process creates a new merged ontology. One aspect that various approaches to ontology merging differ in is the set of criteria they aim to fulfill, i.e., what are the requirements they expect the merged ontology to meet. We have conducted an analysis of the literature and determined which criteria, here called generic merge requirements (GMR), are used by different approaches. Our ultimate aim is to provide a flexible merging approach, where users can actively choose which requirements are important to them, instead of allowing only a very indirect choice by picking the right merging method, something that is not transparent to the user. Unfortunately, not all GMRs are compatible with each other. For instance, one may want to preserve all classes contained in the original ontology in the merged ontology. On the other hand, one could wish to achieve class acyclicity. Likely, these goals conflict. Thus, once a user has chosen which GMRs they consider important, a system is needed that is able to check whether these are compatible and which other requirements can be met simultaneously.

In this paper, we take the first step towards this goal: We analyse the literature to compile a list of GMRs, we provide first insights into their compatibility and we describe a graph-theory based method for determining maximal sets of compatible GMRs.



Completeness
R1. Class preservation
R2. Property preservation
R3. Instance preservation
R4. Correspondence preservation
R5. Correspondences' Property preservation
R6. Property's value preservation
R7. Structure preservation
Logic Properties
Integrity
Minimality
R8. Class redundancy prohibition
R9. Property redundancy prohibition
R10. Instance redundancy prohibition
R11. Extraneous entity prohibition
Deduction
R12. Entailment deduction satisfaction
Constraint
R13. One type restriction
R14. Property value's constraint
R15. Property's domain and range oneness
Model Properties
Acyclicity
R16. Acyclicity in the class hierarchy
R17. Acyclicity in the property hierarchy
R18. Prohibition of properties being inverses of themselves
Connectivity
R19. Unconnected class prohibition
R20. Unconnected property prohibition
Legend
Dimension
Aspect
GMR

Fig. 1: GMR classification

2 GMRs Overview and Classification

To discover the most commonly used GMRs, we have studied and analyzed three different research areas including (i) ontology merging methods [1,2,3,6,7,10], (ii) ontology merging benchmarks [4,9], and (iii) ontology engineering [5,8]. Overall, we classify the GMRs according to three dimensions subdivided into different aspects (see Fig. 1). The three dimensions we identified are:

– **Integrity**: This dimension refers to the degree of knowledge coverage in the merge process via (i) the *completeness* aspect, and to the amount of knowledge redundancy by (ii) the *minimality* aspect.
– **Model Properties**: Within this dimension, the principles of creating a new ontology model are investigated. In this dimension, (i) *acyclicity*, and (ii) *connectivity* satisfaction aspects are taken into account.
– **Logic Properties**: The inference of the expected knowledge with involved constraints is analyzed in this dimension. This includes (i) *deduction*, and (ii) *constraint* satisfaction aspects.

We determined six aspects that GMRs can be grouped into:
Completeness refers to knowledge preservation and coverage:

R1. *Class preservation*: Each class in (all/target) [1] input ontologies should have a mapped class in the merged ontology [1,3,6,7,10].
R2. *Property preservation*: Each property from the (all/target) input ontologies is explicitly in or implied by the merged ontology [7,10].
R3. *Instance preservation*: All instances of (all/target) input ontologies should be preserved in the merged ontology [7,10].
R4. *Correspondence preservation*: If two entities of the input ontologies are corresponding [2], both should map to the same merged entity in the merged ontology [6,7,10].

[1] Preserving all entities from all input ontologies or a preferred one.
[2] This can be equality, similarity or is-a correspondences. In each case, the same type of corresponding should be preserved.

R5. *Correspondences' property preservation*: If any of the corresponding entities from the input ontologies has a certain property, the merged entity should also have this property [6,7].

R6. *Property's value preservation*: Properties' values from the (all/target) input ontologies should be preserved in the merged ontology [6,7]. In case of conflicts a resolution strategy is required.

R7. *Structure preservation*: If two entities are connected via a certain property in an input ontology, their mapped entities in the merged ontology should be connected via the respective mapped property [1], thus preserving the input ontologies' structures in the merged ontology.

Minimality refers to knowledge redundancy and controlling of semantic overlap:

R8. *Class redundancy prohibition*: A class from the (all/target) input ontologies should have at most one mapping in the merged ontology [1,3,4,7,10].

R9. *Property redundancy prohibition*: A property from the (all/target) input ontologies should have at most one mapping in the merged ontology [4].

R10. *Instance redundancy prohibition*: An instance from the (all/target) input ontologies should have at most one mapping in the merged ontology.

R11. *Extraneous entity prohibition*: No additional entities other than input ontologies' entities should be added in the merged result [7].

Deduction refers to the deduction satisfaction with *R12*:

R12. *Entailment deduction satisfaction*: The merged ontology is desirable to be able to entail all entailments of the (all/target) input ontologies [2]. As the semantic consequences of the integration, it can include more entailments but it should at least not miss knowledge from the input ontologies.

Constraint reflects the satisfaction of the ontology constraints:

R13. *One type restriction*: Two corresponding entities should follow the same data type [7]; e.g., if the range of `author_Id` in one of the input ontology is String and in the other one is Integer, then the range of the merged entity `author_Id` in the merged ontology cannot have both types.

R14. *Property value's constraint*: If the (all/target) input ontologies place some restriction on a property's values (e.g., in terms of cardinality or by enumerating possible values) this should be preserved without conflict in the merged ontology [7].

R15. *Property's domain and range oneness*: The merge process should not result in multiple domains or ranges defined for a single property. This rule is recast from the ontology modelling issues in [8].

Acyclicity refers to controlling the chain problem in the merged ontology:

R16. *Acyclicity in the class hierarchy*: A cycle of is-a relationships implies equality of all of the classes in the cycle, since is-a is transitive. Therefore, the merge process should not produce a cycle in the class hierarchy [1,3,5,7,8,10].

R17. *Acyclicity in the property hierarchy*: The merge process should not produce a cycle between properties with respect to the is-subproperty-of relationship [8].

R18. Prohibition of properties being inverses of themselves: The merged process should not cause an inverse recursive definition on the properties [8].

Connectivity refers to the hierarchy connectivity satisfaction:

R19. Unconnected class prohibition: The merge process should not make the classes unconnected [3,7]. Every class that had some connections in the input ontologies before the merge process, should not be unconnected after merge process in the merged ontology.

R20. Unconnected property prohibition: The merge process should not make the properties unconnected [6,8]. Every property that had some connections in the input ontologies before the merge process, should not be unconnected after merge process in the merged ontology.

3 GMR Compatibility

Intuitively, not all GMRs are compatible, e.g., completeness and minimality are difficult to achieve simultaneously. In our ongoing work, we are determining the precise relationships. Typically, users will, however, not be interested in all of them anyhow. Depending on the application they want to create an ontology for, some GMRs might be important to them while others are only nice to have. We therefore propose an approach that takes as input a set of user-selected GMRs, namely $\mathcal{U}$ and an undirected graph $\mathcal{G}$ reflecting the relationship among GMRs. $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices representing the GMRs, and $\mathcal{E}$ is the set of edges. Two GMRs are connected via an edge iff they are compatible.

Given the set $\mathcal{U}$ containing the GMRs the user is interested in, we find the maximum subset of $\mathcal{V}$ containing all vertices out of $\mathcal{U}$ and no incompatible nodes. This may not always be possible, since the user might have chosen incompatible GMRs already. In this case, we search for a maximum subset of $\mathcal{V}$ that preserves as many nodes out of $\mathcal{U}$ as possible and contains compatible nodes only.

Precisely, we recast the problem at hand as maximum clique extraction on $\mathcal{G}$. A clique is a set of fully connected vertices. We thus extract a compatible clique $\mathcal{K}^C$*-Clique*, where $\mathcal{K}$ indicates the number of vertices in the clique, and C denotes that the clique is compatible. $\mathcal{K}^C$*-Clique* includes compatible GMRs from $\mathcal{U}$ and some additional compatible GMRs related to $\mathcal{U}$'s elements. $\mathcal{K}^C$*-max-Clique*, is a clique containing at least $\mathcal{K}$ vertices that is not a subset of any other cliques. To compute the $\mathcal{K}^C$*-max-Clique*, we use the CLIQUES algorithm in [11].

To illustrate our approach with an example, Fig. 2 left shows the graph $\mathcal{G}$, for six GMRs where $\mathcal{U} = \{R_2, R_8, R_{11}\}$. Based on this graph, three different cliques have been extracted, in which the user-selected compatible GMRs, the incompatible $\mathcal{U}$'s elements, and the additional compatible GMRs related to $\mathcal{U}$ are indicated with green, red, and yellow circles, respectively. Two possible represented 3^C*-Clique* are compatible cliques however they are not a maximal clique. The 4^C*-max-Clique* is the best match to the $\mathcal{U}$ and indicates the maximal compatible subset on the sketched $\mathcal{G}$.

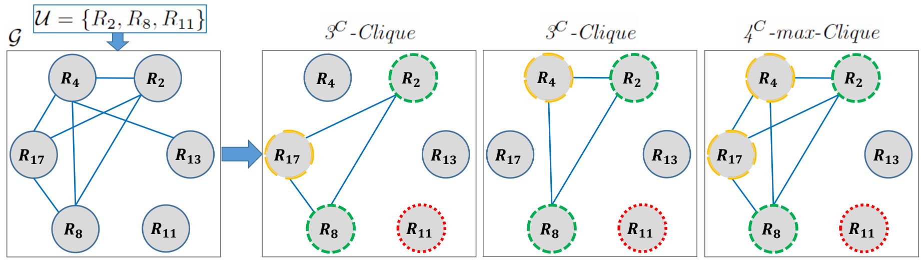


Fig. 2: From left to right: a graph $\mathcal{G}$ for six GMRs; two different compatible 3^C-*Clique*; a compatible 4^C-*max-Clique*.

4 Conclusion

There is a variety of generic merge requirements (GMR) in the ontology engineering domain. We gathered the most common GMRs [3], and classified them into three dimensions. Since not all GMRs can be fulfilled at the same time, we propose an approach that allows users to specify the most important GMRs for their specific task, then determines the maximal compatible subset of GMRs. This result can then be used to select a proper merge method or to parameterize a generic merge method. We are currently working on two aspects of this problem: First, a systematic determination of GMR compatibility and second the development of a generic, parameterizable merge method.

Acknowledgments. S. Babalou is supported by a scholarship from German Academic Exchange Service (DAAD).

References

1. F. Duchateau and Z. Bellahsene. Measuring the quality of an integrated schema. In *ER*, pages 261–273, 2010.
2. E. Jiménez-Ruiz, B. C. Grau, I. Horrocks, and R. Berlanga. Ontology integration using mappings: Towards getting the right logical consequences. In *ESWC*, 2009.
3. S. P. Ju, H. E. Esquivel, A. M. Rebollar, and M. C. Su. Creado–a methodology to create domain ontologies using parameter-based ontology merging techniques. In *MICAI*, 2011.
4. M. Mahfoudh, G. Forestier, and M. Hassenforder. A benchmark for ontologies merging assessment. In *KSEM*, pages 555–566, 2016.
5. N. F. Noy, D. L. McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.
6. N. F. Noy and M. A. Musen. The prompt suite: interactive tools for ontology merging and mapping. *IJHCS*, 59(6):983–1024, 2003.
7. R. A. Pottinger and P. A. Bernstein. Merging models based on given correspondences. In *VLDB Endowment*, 2003.
8. M. Poveda-Villalón, M. C. Suárez-Figueroa, and A. Gómez-Pérez. Validating ontologies with oops! pages 267–281, 2012.
9. S. Raunich and E. Rahm. Towards a benchmark for ontology merging. In *OTM*, volume 7567, pages 124–133, 2012.
10. S. Raunich and E. Rahm. Target-driven merging of taxonomies with atom. *Inf. Syst.*, 42:1–14, 2014.
11. E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *TCS*, 2006.

[3] see also: http://comerger.uni-jena.de/requirement.jsp