Ontology Merging using Answer Set Programming and Linguistic Knowledge

Jürgen Bock[1], Rodney Topor[2], and Raphael Volz[1]

[1] FZI Forschungszentrum Informatik, Karlsruhe, Germany
{bock, volz}@fzi.de
[2] Griffith University, Brisbane, Australia
r.topor@griffith.edu.au

Abstract. With the increasing number of ontologies available on the web, the problem of merging ontologies from different sources to interoperate applications becomes important. This paper presents a novel approach for merging of light-weight ontologies based on answer set programming (ASP) and linguistic background knowledge. ASP provides a declarative execution environment for intuitive merging rules. WordNet provides broad linguistic knowledge that is used to identify corresponding concepts. We present a semi-automatic merging algorithm, where users can choose appropriate results from a set of suggestions.

1 Introduction

Many semantics-based applications are isolated applications utilising single ontologies to improve data access and navigation.

The popularity of the Web 2.0 theme has brought high attention to so-called "mashups", where data from several applications is combined to provide novel applications. For example, `housingmaps.com` combines map data with real estate information to depict houses for sale on a map. Beyond the Web, ontology merging is also a fundamental task in enterprise data integration. In many cases data interoperation goes beyond mapping-based translation of data between applications and a recent trend in large enterprises is to create meta-databases which holds master data about the schemas (or ontologies) of all applications in the enterprise.

This paper[3] addresses ontology merging, *i.e.* creating a single, coherent ontology out of several different ones, and presents a novel approach for merging of light-weight ontologies. We present a semi-automatic merging algorithm, where users can choose appropriate results from a set of suggestions. Our techniques are based on combining answer set programming (ASP) with linguistic background knowledge, which brings several benefits for ontology merging:

[3] We thank Kewen Wang for useful comments and Roman Schindlauer, the main developer of dlvhex, for useful information and fruitful web chats. Early discussions with Marilyn Ford have been very helpful to identify this project.

– Correspondence detection: Linguistic knowledge is used to detect correspondences between concepts based on synonymy
– Merging options: ASP calculates several answer sets which provide merging options among which the user can choose
– Extensibility: We provide intuitive merging rules that can be easily extended to capture domain-specific extensions

While our general approach is not language specific, our implementation currently deals with WordNet [1] as a basic broad linguistic resource that is used to identify corresponding concepts. Due to our observation on the prevalence of light-weight ontologies, we concentrate on merging light-weight ontologies based on the recently proposed SKOS standard for controlled vocabularies.

The problem of ontology merging has been addressed by several authors and the use of linguistic information is common in these approaches.

The PROMPT Suite [2] is a collection of tools, available as a plugin for the Protégé ontology editor. It incorporates lexical and (optionally) linguistic knowledge to identify similar or synonymous entities. However, the PROMPT approach utilisies linguistic resources only to a limited extent and requires a high degree of user-interaction.

Ehrig and Sure [3] suggest 17 rules to gain similarity measures between a number of ontologies to be mapped. Since the rules were "manually formulated by domain experts", this approach shows how explicit encoding of intuitive rules works well for the ontology merging task.

Wang *et al.* [4] use ASP for ontology merging and alignment of expressive DL-programs [5]. The paper focuses solely on conflict resolution and maintaining of consistency throughout the merging/alignment process. We pick up the idea of using ASP, but follow a more practical approach.

2 Answer Set Programming

We choose Answer Set Programming (ASP) [6] as implementation language for several reasons. Firstly, it allows very compact encodings for complex problems, such as the graph colouring problem. Secondly, it is a purely declarative programming paradigm, which allows to formulate the problem in terms of "what" should be done, instead of "how" to compute the solution.

Intuitively, ASP programs are a set of three basic constructs: facts (*e.g.*: $light$), rules (*e.g.*: $light \leftarrow switchOn$), and constraints (*e.g.*: $\leftarrow light, daytime$). With particular respect to DLV, atoms can be default (not $light$) or classically negated ($\neg light$), and disjunction can occur in rule heads. Refer to Baral [7] for a formal account.

Using dlvhex as an extension to DLV, we were able to provide an external atom to access the WordNet database[4]. The external atom was designed to be useful for a wider range of applications beyond the scope of this merging algorithm.

[4] `http://con.fusion.at/dlvhex/download.php`

3 Ontology Merging using ASP and Linguistic Knowledge

The fundamental assumption behind our algorithm is to provide a semi-automatic merging approach, which presents a number of different possible merging solutions to the user, who can finally choose the one(s) best suitable for her needs. The ontology merging algorithm itself is designed to follow intuitive rules to make it not only easy to understand but also extensible. The rule set incorporates information given by the structure of the ontologies to be merged, as well as additional linguistic background knowledge and respects the following issues:

- Exploit *linguistic information*, since ontologies typically follow a human knowledge model and entities are labeled in natural language terms.
- Allow or forbid different *ontology structures*, such as trees or DAGs.
- Respect *explicit domain knowledge* provided by the user via certain flags or parameters.
- Provide an option for *brave merging* to further reduce the number of merging suggestions.

3.1 Formal Design

For two concepts c_1 and c_2, by $c_1 \simeq c_2$ we denote that their labels are identical or synonyms. $c_1 \prec c_2$ denotes that a label of c_1 is linguistically narrower (*i.e.* a direct hyponym or meronym in this approach) of a label of c_2. By $c_1 \leq_C c_2$ we denote that c_1 is defined narrower than c_2 in one of the input ontologies.

Concept Melding. Intuitively, two concepts of different ontologies can be melded, if any of their labels are identical or synonyms[5].

$$meld(c_1, c_2) \ \vee \ \neg meld(c_1, c_2) \quad \leftarrow \quad c_1 \simeq c_2 \tag{1}$$

$$\leftarrow \quad c_1 \simeq c_2, \ \mathsf{not}\ c_1 \prec c_2, \ \mathsf{not}\ c_2 \prec c_1, \ \neg meld(c_1, c_2) \tag{2}$$

Since some words can be synonyms, as well as in a linguistical narrower relation, they do not necessarily have to be melded. This can be expressed by rule (1). However, if c_1 and c_2 are not in a linguistic narrowing relation, they must be melded, which will be forced by constraint (2).

Hierarchy Restructuring. Intuitively, two concepts can be merged in a potential narrowing relation, if they are in a narrowing relation in one of the input ontologies, or if they are in different ontologies but in a linguistical narrowing relation.

[5] The rules presented in this section are denoted in a formal and simplified way to demonstrate basic ideas of the algorithm. They violate rule safety and other restrictions and cannot be implemented straightforwardly. Definitions of auxiliary atoms are omitted. Please refer to [8] for the full translation to the implemented set of safe rules.

For ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ and $i, j \in \{1, 2\}$

$$pot\_narr(c_2, c_1) \leftarrow c_1 \leq_C c_2 \quad (c_1, c_2 \in \mathcal{O}_i) \tag{3}$$
$$pot\_narr(c_2, c_1) \leftarrow c_1 \prec c_2 \quad (c_1 \in \mathcal{O}_i,\ c_2 \in \mathcal{O}_j,\ i \neq j) \tag{4}$$

$$m\_narr(c_1, c_2) \vee \neg m\_narr(c_1, c_2) \leftarrow pot\_narr(c_1, c_2) \tag{5}$$

Firstly, narrowing relations given by the input ontologies and linguistic information are collected in rules (3) and (4). (Without loss of generality, only atomic concepts are considered in these rules, *i.e.*, concepts that are not yet melded.) However, a reasonable merging will not contain all potential narrowing relations due to transitivity and adjustable restrictions to the final structure (see later in this section). Therefore, a second step identifies appropriate subsets of all potential narrowing relations to form the different merging proposals. The main idea is, to either pick a potential narrowing relation for the final merging, or not (rule (5)).

3.2 User Guidance

To restrict the (so far exponential) number of possible merging solutions to reasonable ones, constraints are used, which are enabled by several flags, set by the user, namely *no singles*, *one root*, *single parent*, and *always meld*.

Let $b(c)$ and $n(c)$ be atoms containing all concepts c that are chosen to be merged as a broader ($b(c)$) or narrower ($n(c)$) concept to any other. A root concept $root(c)$ is defined as a concept that is broader but not narrower than any other concept, or an isolated concept. Let $chained(c, d)$ be the transitive closure of potential narrowing relations that form a chain of at least three concepts.

The following constraints are enabled by the according flags:

$$\leftarrow \text{ not } b(c),\ \text{not } n(c),\ no\_singles \tag{6}$$
$$\leftarrow root(c),\ root(d),\ c \neq d,\ one\_root \tag{7}$$
$$\leftarrow m\_narr(c, e),\ m\_narr(d, e),\ c \neq d,\ single\_parent \tag{8}$$
$$\leftarrow \neg meld(c_1, c_2),\ c_1 \simeq c_2,\ always\_meld \tag{9}$$
$$\leftarrow m\_narr(c, d),\ pot\_narr(c, d),\ chained(c, d),\ brave \tag{10}$$

The *no singles* constraint (6) does not allow concepts that do not occur as either a broader, or a narrower concept. The *one root* restriction (7) constrains the possible merging solutions to those where only one root concept exists. Note that *one root* implies *no singles*, since every single node is a root. Ontologies, that are organised in a tree structure require concepts to be connected only by one single incoming narrowing relation. This can be enforced by setting the *single parent* flag, which enables constraint (8). The *always meld* constraint (9) disables the generation of multiple answer set by simple disallowing meldable concepts not to be melded. This can reduce the number of merging suggestion drastically. The *brave* constraint (10) makes the algorithm greedy in terms of preferring a more nested hierarchy to a flatter one. This is achieved by preferring longer chains of narrowing relations to shortcuts of a single narrowing edge.

4 Conclusion

We have presented a novel approach to ontology merging, incorporating natural language background knowledge and a direct implementation of intuitive merging rules. For the algorithm, a number of requirements have been identified, and formally translated into declarative rules for ASP. This formal design could be implemented straightforwardly in only 56 logical lines of code (cf. [8]) for dlvhex, which we extended by a new external plug-in to deal with WordNet. The algorithm addresses the two main aspects of concept melding and hierarchy restructuring. A possible merging contains a reasonable subset of these potential narrowing relations by constraining possible solutions according to a number of (adjustable) conditions.

This work focused mainly on providing a new methodology of computing ontology mergings, rather than producing an off-the-shelf application.

Early experiments have shown promising results for various flag combinations, and further evaluations will be conducted. Future work also includes the extension and refinement of the algorithm, such as the use of weak constraints and aggregates. These would allow for extensions, such as checking for multi-level linguistic relations, ordering answer sets and discarding answer sets below a certain confidence threshold, or the use of the linguistic resource also to augment the merging results by missing concepts. Furthermore, a modularisation of the algorithm would allow for user interaction for crucial decisions in disjunctive rules.

References

1. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM **38**(11) (1995) 39–41
2. Noy, N.F., Musen, M.A.: The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping. Int. J. Hum.-Comput. Stud. **59**(6) (2003) 983–1024
3. Ehrig, M., Sure, Y.: Ontology Mapping - An Integrated Approach. In: Proc. 1st European Semantic Web Synopsium (ESWS). (2004) 76–91
4. Wang, K., Antoniou, G., Topor, R.W., Sattar, A.: Merging and Aligning Ontologies in dl-Programs. In: Proc. 1st International Conference on Rules and Rule Markup Languages for the Semantic Web. (2005) 160–171
5. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining Answer Set Programming with Description Logics for the Semantic Web. Technical Report INFSYS RR-1843-03-13, Institut für Informationssysteme, Technische Universität Wien, A-1040 Vienna, Austria (December 2003)
6. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing **9**(3–4) (1991) 365–386
7. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
8. Bock, J.: Ontology Merging using Answer Set Programming and WordNet. Honours Thesis, Griffith University (October 2006)