Partial Merging of Semi-structured Knowledgebases

Ladislau Bölöni and Damla Turgut

Department of Electrical and Computer Engineering
University of Central Florida
Orlando, Florida, 32816
{lboloni,turgut}@cpe.ucf.edu

Abstract. Automatizing the merging of knowledgebases is an important step towards more efficient knowledge management. The cases when two knowledgebases need to be merged completely into a monolithic result, however, are relatively rare. Most often, some of the information is irrelevant, not trusted, or needs special treatment as a belief, opinion or preference. This paper presents an approach for partial merging of semi-structured knowledgebases. The merging scheme is based on the partitioning of the knowledgebases through the use of *swimlines* and the application of specific primitive merging algorithms in the partitions thus created. This approach allows the participants of the merging operation to specify their intentions in the merging process in an efficient and intuitive way.

1 Introduction

Knowledge management systems are frequently storing information in semi-structured knowledgebases. These contain a mix of formal and informal information. Knowledge sharing is one of the key elements of a knowledge management system (in addition to knowledge discovery, capture and application [1-3]). Despite this, knowledge sharing is only minimally automatized across organizations. The reasons are both technical and human. During knowledge capture, companies can enforce the use of a single knowledgebase. More often than not, however, multiple independent knowledgebases with restricted access are set up. Transfer of information between the knowledgebases is typically happening through human interaction: meetings, water-cooler discussion etc. In these interactions there is a well-controlled flow of information. Decision to share a piece of information is sometimes the result of long deliberation, involving the potential beneficial or negative effects of the disclosure. Received information is not accepted uncritically, but it can be stored as tentative information, or reformulated as second order knowledge (e.g. knowledge about someone's opinion about a subject).This article presents a way of specifying partial merges through the use of *swimlines* and succinctly describe several use cases and the resulting algorithms.

2 Knowledge Base Organization. Swimlines

Sharing knowledge is an important part of the knowledge management of researchers. Examples of knowledge sharing are making presentations, distributing documents or informal conversations. The key difficulties in any knowledge sharing process is determining what to share (with the associated privacy issues), and how the shared information is going to *merged* into the receivers knowledgebase.

Let us consider the example of a researcher working at a university. Researchers frequently share results with the public-at-large. One the other hand, some data on the ongoing work is only shared with members of the group or collaborators. Some data is protected by nondisclosure agreements (NDA's), are military secrets or company contracts. There are other situations when the data can be confidential: students grades, paper reviews, etc. The main challenge is to design of a privacy scheme which is rich enough to handle the complexity of the privacy and trust relationships, and at the same time simple and understandable enough such that it can be adopted for everyday use.

We present the organization of the knowledgebase of Kraken, a knowledge management system developed by our group. The general structure of the Kraken knowledgebase is a flat collection of *entries*, each of them represented by a top-level unique resource identifier (URI). The organization of an entry is shown in Fig. 1. There are two distinct parts: the *content* and the *metadata*. The content of the entry the unstructured part of the knowledgebase: an arbitrary collection of documents, in their native format. The metadata of an entry is a set of RDF triples, divided into *chunks*. A chunk represents an aspect of the entry, examples being bibliographic information, calendaring information, notes, summaries and so on.

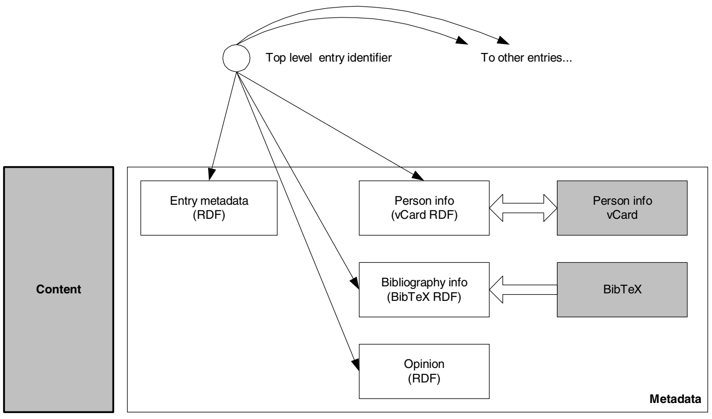

Fig. 1. The structure of the kraken entry

The valid format of the chunks is described by associated ontologies written in the OWL Lite subset of the OWL ontology [4]. For many chunks, there can be one or two representations in *external formats* as well. These are usually legacy representations of the given aspect of the data entry. Thus, for every chunk of data, there is an *internal representation* in the Kraken (always RDF), a *primary representation* that is used for the editing of the data, and can have several external formats. The primary representation might or might not be the same as the internal one. In order of a representation R to be accepted, Kraken needs to have at least a converter from the primary representation P to R.

2.1 A Swimline Based Data Privacy Model

We define a *swimline* σ as a boolean function which separates private from public data.

$$\sigma : KB \rightarrow \{0, 1\}$$

The usual interpretation is that σ *((s,p,o))=1* if the RDF triplet *(s,p,o)* is visible, while σ *((s,p,o))=0* indicates that the triplet is hidden for the purpose of a transaction. We are especially interested in *well-formed swimlines*, where the visible part of the knowledgebase represents a valid knowledgebase, maintaining the same set of constraints as the original knowledgebase.

To be well formed, for the kraken data model, the swimline is always separating complete-chunks of data:

$$\forall (s_1, p_1, o_1), (s_2, p_2, o_2) \in C \subset KB \Rightarrow \sigma(s_1, p_1, o_1) = \sigma(s_2, p_2, o_2)$$

This also implies that if a chunk is public, its external format variants are also public, and conversely, a private chunk remains private in its external formats as well. In addition, if the top level of an entry is not visible, then the rest of the entry is hidden as well.

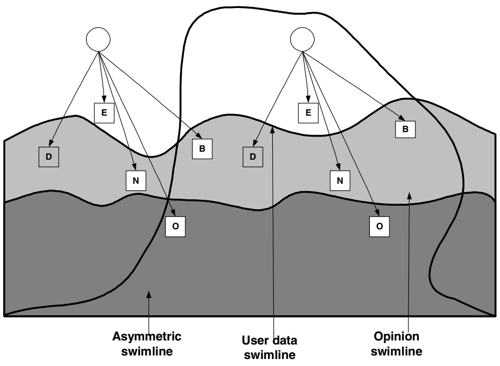

Fig. 2. The swimline data privacy model

Figure 2 shows a knowledgebase with three swimlines. The opinion swimline declares as private only the opinion chuncks, while the rest of the data is public. The user data swimline, on the other hand declares public only the basic entry data. These swimlines are symmetric, they are using an identical policy for every entry. Although this is an appropriate choice for system-wide swimlines, and they can be described concisely, they are not the only possible choice. The user data swimline in Figure 2, for example, used different policies for the represented entries.

We define the union and intersection of swimlines as the conjunction and disjunction of the respective swimline functions. The negation of swimlines, however, in case of the Kraken data model, does not always give a well formed swimline.

3 Partial Merging of Knowledgebases

For the purposes of the following discussion, we define an act of knowledge sharing as set of changes performed in the *knowledge receiver*, which are determined by the *knowledge source*'s knowledgebase and the *merging scheme* used. Merging schemes are a combination of primitive merging algorithms and partitioning of the knowledgebases through swimlines.

We propose the following set of primitive merging algorithms:

- **No merge (NM):** The receiver's knowledgebase will be unchanged.
- **Entry Overwrite (EO):** Entries are matched against each other. The source entry completely replaces the receiver entries.
- **Chunk Overwrite (CO):** Entries are matched. Whenever a chunk exists in the source entry, it will completely overwrite the corresponding chunk in receiver. Chunks in the receiver which do not exist in the source are not modified.
- **Property Overwrite (PO):** Entries, chunks and properties are matched against each other. If a property exists in the source, it will overwrite the corresponding property in the receiver. Properties which do not exist in the source but exist in the receiver are not modified.
- **Entry Reference (ER):** Entries are matched. If an entry exists both in the source and the receiver, the source entry is copied as a *chunk* of the receiver and labeled with the identifier of the source. If the entry does not exist in the receiver, an empty entry is created with the same identifier as in the source, and the source entry attached as a chunk.
- **Chunk Reference (CR):** Entries are matched. If a chunk exists in the source, it will be copied to the receiver, and labeled with the identifier of the source. For an illustration of the application of the ER and CR merging primitives see Figure 3.

The primitive merging algorithms assume the existence of a *matching algorithm* which associates entries and chunks which represent the same knowledge entity or aspect in the different knowledgebases. The algorithm currently used by us is based on the identity of the URI's and an *identity table* which contains a set of **owl-sameAs** relations. Chunks and properties are always matched by name.

A *merging scheme* is a combination of swimlines, merging algorithms and a single matching algorithm. Swimlines can be contributed both by the knowledge source and knowledge destination. The set of swimlines in a merging scheme divides the merging scheme into domains. Every domain is characterized by a merging algorithm.

The final purpose of the merging scheme is to satisfy the *intent* of the participants of the communication. The communicators express the intent in the form of swimlines. The motivations of the choice of particular swimlines can be different: willingness to expose information, trust in its own data, trust in the communication partners data, trust in the communication partners assessment of its own data and so forth.

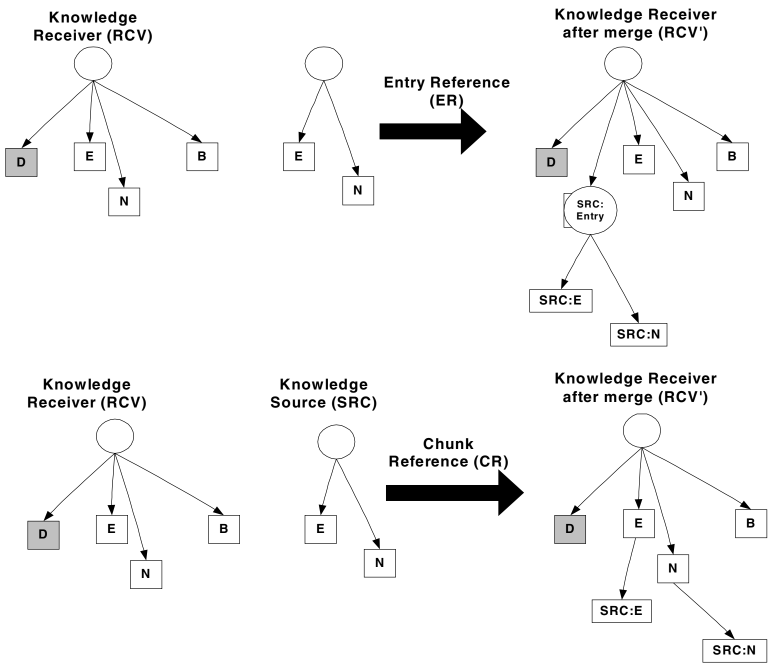

Fig. 3. Entry Reference (ER) and Chunk Reference (CR) merging primitives

Let us now proceed to examples illustrating how the participants in a knowledge sharing operation can accomplish their sharing intentions through the use of swimlines.

In our first, simplest example, the knowledge source provides a single, visibility swimline. The knowledge receiver also provides a single *data protection* swimline. The intention of the receiver is to maintain the data below the protection swimline unchanged.

The merge can still happen through reference merging algorithms, which do not modify existing data (for example, CR). The data above the data protection swimline can be modified, and a algorithm such as property merge applied. The resulting merging scheme is presented in Figure 4a.

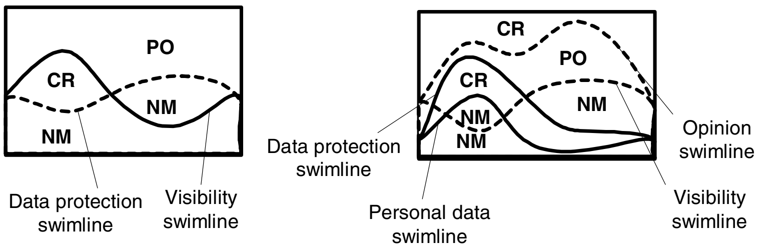

Fig. 4. Two examples of merging schemas determined by swimlines

In our second example, both the knowledge source and the receiver provide two swimlines each. The knowledge source, in addition to the visibility swimline also provides an *opinion swimline*, where the information below that swimline is seen as personal opinion, and requests to be treated as such. The receiver also provides two swimlines, the data protection swimline, and the *definite knowledge* swimline. The receiver considers that it has definite, final knowledge on the data below that swimline, and it is not interested in new information regarding those aspects. A merging scheme handling the semantic implications of these swimlines is presented in the Figure 4b.

4 Related Work

A number of projects proposed ontology merging tools and algorithms, the main differentiating factor being (a) whether they act at the ontology or the knowledgebase level and (b) in the amount of user intervention required. In OBSERVER [6], interoperation across ontologies is achieved by traversing semantic relationships defined between terms across ontologies and its architecture is designed for query processing in a global information system. ONION [7] represents ontologies in a graph-oriented model with a small algebraic set to facilitate automatic composition. Formal Concept Analysis is performed on instances of extracted language processing outputs from a domain specific set of texts to form a suitable ontology in FICA-Merge [8]. PROMPT [9] provides a semi-automatic approach to merging ontologies and is designed to work with a frame-based knowledge model. Chimaera [10] is a browser-based editing and merging tool for creating and maintaining ontologies. The swimline model is positioned as a more streamlined way to specify the merging rules, although in practical situations, a user might consider using it together with a more fine grained tool such as PROMPT or Chimaera.

5 Conclusions

This paper presented an approach for merging semi-structured algorithms based on the concept of swimlines. We presented how relatively complex, customized knowledge sharing operations can be presented through a combination of swimlines and primitive merging algorithms. This model was implemented in the Kraken knowledge management system.

Significant theoretical and practical challenges remain. From the theoretical point of view, the properties of the merging schemes need to be investigated: under what conditions is the merging scheme idempotent, associative, stable? How can we avoid the explosion of the size of the knowledgebases after repeated reference merging operations? How can the knowledge sharing operations be extended to multiple participants?

References

1. Nonaka., I.: A dynamic theory of organizational knowledge creation. Organizational Science **5** (1994) 14-37
2. Grant, R.: Prospering in dynamically competitive environments: Organizational capabilities as knowledge integration. Organizational Science **7** (1996) 85--94
3. Grant, R.: Towards a knowledge-based theory of the firm. Strategic Management Journal **17** (1996) 375-387
4. Owl web ontology language reference. URL http://www.w3.org/TR/owl-ref/ (2003)
5. Pottinger, R.A., Bernstein, P.A.: Merging models based on given correspondences. In Proceedings of the 29th International Conference on Very Large Databases. (2003) 862-873
6. E. Mena V. Kashyap, A. P. Sheth, and A. Illarramendi, OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In Conference on Cooperative Information Systems, 1996, pp. 14–25
7. P. Mitra, G. Wiederhold, and M. Kersten, A graph-oriented model for articulation of ontology interdependencies, Lecture Notes in Computer Science, vol. 1777, pp. 86+, 2000.
8. G. Stumme and A. Maedche, FCA-MERGE: Bottom-up merging of ontologies. In *IJCAI*, 2001, pp. 225–234.
9. N.F. Noy and M. A. Musen, PROMPT: Algorithm and tool for automated ontology merging and alignment. In AAAI/IAAI, 2000, pp. 450–455.
10. D. McGuinnes, R. Fikes, J. Rice, S. Wilder, An environment for merging and testing large ontologies. In Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000), 2000.