Domain Knowledge Articulation using Integration Graphs

Madalina Croitoru Ernesto Compatangelo

Department of Computing Science, University of Aberdeen, UK
{mcroitor, compatan}@csd.abdn.ac.uk

Abstract

In order to be queried and reused, different domain knowledge bases must be shared using the same representation. In this paper, we introduce a graph-based knowledge model to articulate independently developed ontologies that describe the same application domain. Our knowledge representation formalism is based on a new type of graphs denoted as "hierarchical graphs". This representation provides a basis for the heuristic inferences and for the logical deductions that are used to detect semantic mismatches between domain knowledge bases whenever they must be composed. Integration graphs can also be viewed as a suitable data modelling formalism for information integration.

1 Introduction

Domain Knowledge describes the matter of interest in some context, such as entities and their relationships in a conceptual schema or taxonomies of shared concepts in an ontology. In any case, domain knowledge is considered separately from the problems or tasks that may arise in its application context [11]. In recent years, the problem of establishing links between different Domain Knowledge Bases (DKBs for short) has received considerable attention from both database and knowledge engineering communities [2, 7]. Specifically, the problem of creating a shared view that links two (or more) DKBs without modifying their respective contents is being actively investigated. The introduction of a shared view (denoted as an *articulation*) is considered as an effective way of enabling both queries to distributed data and knowledge bases and ontology sharing & reuse [9, 5]

In order to be queried and reused, different domain knowledge bases must be shared using the same representation. In other words, *knowledge interoperability* must be enabled first. In the past decade, research in Knowledge Representation (KR) led to the introduction of several KR languages and systems. These are based on diverse knowledge models, such as semantic networks, frames, description logics, and conceptual graphs, which are outlined below.

- Semantic networks represent knowledge as labelled direct graphs, where nodes denote concepts, and arcs denote the various relations between concepts [10].

- Frames represent knowledge (e.g. individuals, classes or general concepts) using an object-like structure with attached properties. Frames are typically arranged in a taxonomic hierarchy where each of them is linked to one (or more) parent frames [6].

- Description Logics (DLs) represent knowledge similarly to frames, explicitly distinguishing between a TBox, and an ABox. While the former describes classes and their mutual links, the latter describes individual elements possibly belonging to classes explicitly defined in the TBox. The set theoretic semantics of DLs allows the introduction of specialised deductive services such as DKB consistency, concept normalisation and completion, subsumption between concepts, and instance retrieval [1].

- Conceptual graphs represent knowledge as bipartite labelled graphs where nodes denote entities or relationships them. The knowledge model underpinning conceptual graphs is logically sound and complete w.r.t. the semantics of First-Order Predicate Logics (FOPL). Reasoning about conceptual graphs is based on graph-theoretical operations that rely on graph homomorphism [10].

Despite the variety of available knowledge models, the representation of knowledge that specifies mappings between related domain knowledge bases is still an open research problem. Moreover, the complex structure of these mappings affects the development of automated reasoning algorithms to analyse and validate them. More specifically:

- Semantic Networks suffer from the lack of a clear formal semantics, even if some approaches versions are formally defined within the framework of FOPL. Even in these approaches, the expressive power of Semantic Networks is not adequate to represent the structure of mappings between related concepts in partially overlapping DKBs.

- The semantics of frames is not completely formalised, which prevents non-trivial deductions about them. Moreover, the object-oriented, rigid structure of frames is not particularly suited to express mappings between concepts.

- Despite their fully defined set-theoretic semantics, DLs suffer from the same drawbacks of frames, which is due to their rigid structure. However, articulation-specific deductions can be performed in a DL-based framework [5]. Nevertheless, the terminological nature of these deductions may prevent the detection of links between semantically related concepts, individuals or properties expressed in a structurally different way.

- Although conceptual graphs benefit from a solid logic foundation, their expressive power may not be completely adequate to express mappings. An extension that can capture the hierarchical aspect of articulation modelling might overcome this drawbacks.

Graph-based models have advantages over frame-based models in expressing certain kinds of mappings (e.g. mapping properties into concepts and vice versa). However, their expressiveness might not be adequate for modelling the DKBs to be articulated, integrated or merged, and their automated deductive capabilities are rather limited. In order to overcome the above drawbacks, in this paper we propose *integration graphs*, which combine the advantages of graph-based models with those of frame-based models. Integration graphs are hierarchical graphs used to articulate independently developed ontologies that describe the same application domain. The representation formalism of integration graphs provides a basis for the logical inferences needed to compose domain knowledge bases and to detect semantic mismatches between them. Integration graphs can also be viewed as a suitable data modelling formalism for information integration systems by allowing the end user to restrict their queries either to any individual sources or to their shared view. Therefore, our graph-based model can prove internal navigation instructions to support the query execution plans needed to interact with the various knowledge sources.

2 Using integration graphs to articulate independent sources

Knowledge integration is the task of combining different domain knowledge bases. In order to accomplish this task, first of all a suitable representation must be used. Secondly, a mechanism for combining the knowledge must be developed. In this paper a graphical formalism, denoted as integration graphs, is used to describe knowledge combination (e.g. articulation, integration, alignment). Integration graphs describe the knowledge in terms of nodes. The links between similar concepts in different knowledge domains are described in terms of edges. Integration graphs provide good navigational properties, as well as a logic foundation that allows automated semantic reasoning.

2.1 The domain knowledge articulation process

Knowledge interoperability explicitly depends on the adopted representation. However, the choice of a common knowledge model (and thus of the same representation) does not automatically imply that knowledge in different DKBs can be straightforwardly shared and reused. Another fundamental operation must be done to enable knowledge interoperability: all the diverse (but partially overlapping) DKBs must be "aligned", i.e. they must reach a condition of mutual agreement. This can be done by *articulating* them.

The introduction of an articulation establishes semantic bridges between concepts in the intersection of two partially overlapping DKBs, which remain otherwise independent and physically separated. An articulation creates a shared view, where each concept in the view is mapped into one or more concepts in the first DKB *and* into one or more concepts in the second DKB. The participating domain knowledge bases are completely unaffected by the introduction of this view [5]. A graphical overview of the articulation of two domain knowledge bases is shown in Figure 1, where the dashed lines connect each concept in the articulation to a couple of related concepts, one in each source DKB. Hierarchical relationships between

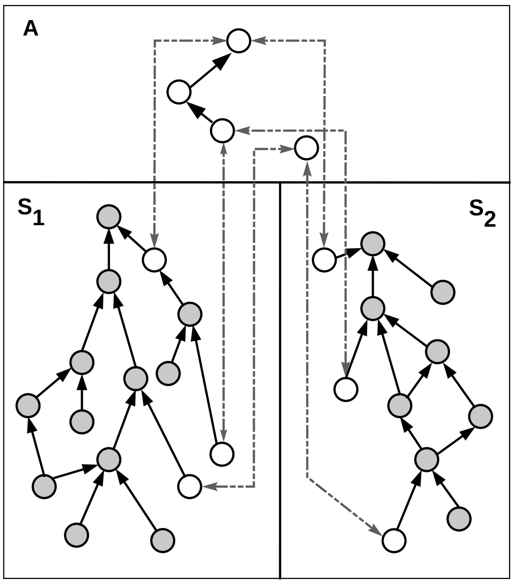

Figure 1: Articulation of DKBs S_1 and S_2, giving rise to the shared view A

articulation concepts can also be defined. Note that the introduction of an articulation schema does not modify the schemas to be shared in any way. Moreover, an articulation considers all participating DKBs as “peers”. This means that it does not implicit distinguish between a “more general” DKB, which should not be modified in any case, and a “more specific” DKB which may need to be modified in order to align it to the other one.

Following a recent approach to the articulation of conceptual schemas and ontologies [8, 9], a graphical representation can be used to articulate two (or more) DKBs. The semantic bridges between their concepts can be modelled using both logical rules (e.g., semantic implication between terms across ontologies) and functional rules (e.g. dealing with conversion functions between terms across ontologies). The articulation rules indicate which terms, individually or in conjunction, are related in the source ontologies. Articulation ontology contains these terms and the relationships between them. The term articulation refers to the articulation ontology and to the set of mappings that relate elements in the articulation to terms in the source ontologies. A unified ontology consists of the source ontologies integrated using the articulation.

2.2 Supporting domain knowledge articulation with integration graphs

Integration graphs are a graphical representation for articulating domain knowledge bases. They formalise the integration process using hierarchical graphs [3]. Nodes in integration graphs describe domain concepts. Simple nodes denote new concepts, while complex nodes denote hidden, previously known ontologies. Each edge linking a simple node and a complex node has an articulation rule associated to it and instructions that describe its use.

In order to illustrate the notion of integration graph, we consider a simplified version of an example originally introduced in [9]. The two sources graphs *Carrier* and *Factory* in Figure 2 represent two sources ontologies. A possible articulation of these ontologies is described by the integration graph *Transportation* in Figure 3.

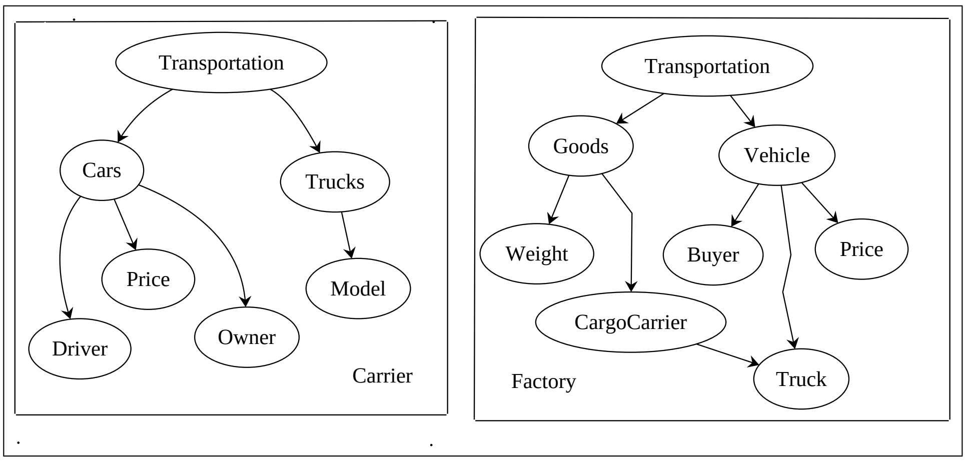

Figure 2: The graphs describing two ontologies, Carrier and Factory

The articulation rules H1–H4 (see Figure 4) describe the edges that connect simple nodes (represented by circles or ovals) to complex nodes (represented by squares or rectangles) in the integration graph. A simple node denotes a new concept, while a complex node denotes a

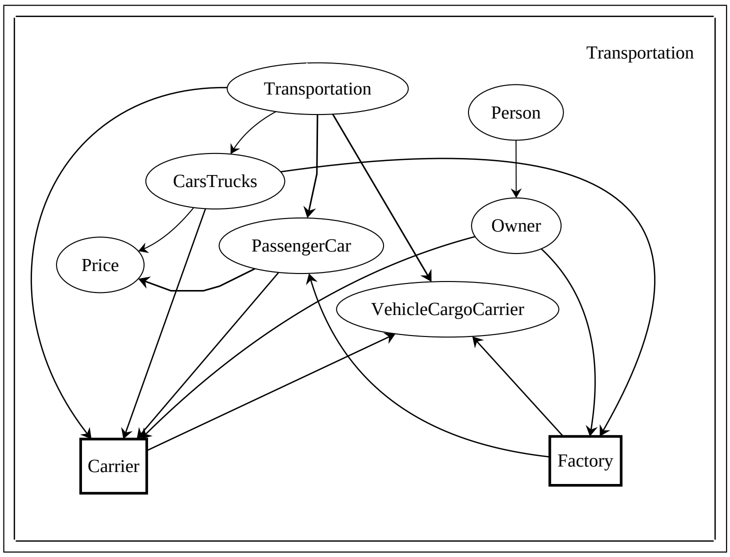

Figure 3: The articulation ontology Transportation

known ontology (the source one in our example). Therefore, each edge linking a simple node to a complex one has an articulation rule associated to it, as well as instructions which describe its usage. In our example, all (simple) nodes are assumed to be visible. Each edge with (at least) one complex node at its extremity has both an associated a rule Hi and substitutions that explain which simple node (inside a complex one) is involved. The edges used in our example (where name ambiguities are solved by qualifications) are shown in Figure 4.

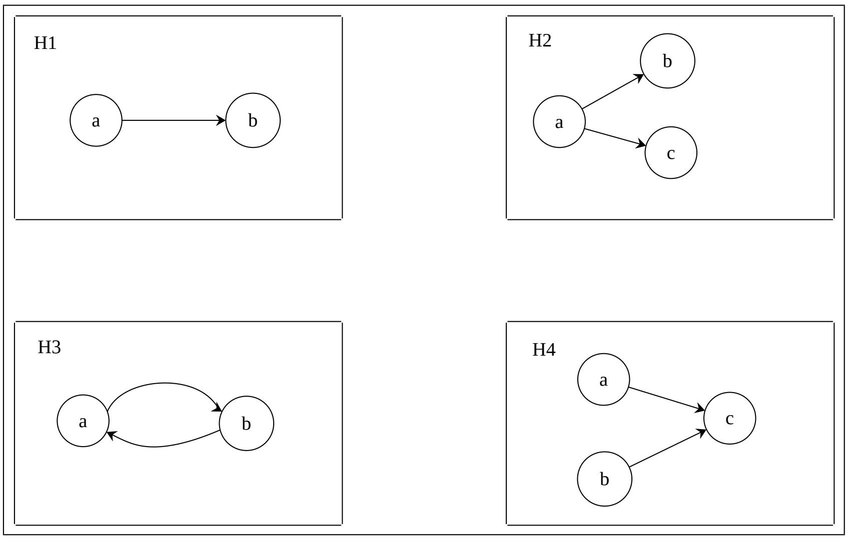

Figure 4: The articulation rules used in Transportation graph

The specification of the articulation rules in Figure 3 is presented in Table 1. This model offers a favourable approach when dealing with navigational problems inside a graph. It also gives the possibility of a logic based formalism for reasoning.

Transportation $\longrightarrow$ **Carrier**:
H1; a$\longrightarrow$ T.Transportation ; b$\longrightarrow$ C.Transportation
CarsTrucks$\longrightarrow$ **Carrier** :
H2; a$\longrightarrow$ T.CarsTrucks, b$\longrightarrow$ C.Cars, c$\longrightarrow$ C.Trucks
PassengerCar$\longrightarrow$ **Carrier** : H1; a$\longrightarrow$PassengerCar, b$\longrightarrow$ C.Cars
Owner$\longrightarrow$ **Carrier** : H3;a$\longrightarrow$ T.Owner, b$\longrightarrow$ C.Owner
CarTrucks$\longrightarrow$ **Factory** : H1; a$\longrightarrow$ CarTrucks, b$\longrightarrow$ Vehicle
Owner $\longrightarrow$ **Factory** : H3;a$\longrightarrow$ T.Owner, b$\longrightarrow$ Buyer
Factory$\longrightarrow$ **PassengerCar** : H1; a$\longrightarrow$ Vehicle, b$\longrightarrow$ PassengerCar
Carrier$\longrightarrow$ **VehicleCargoCarrier** :
H1; a$\longrightarrow$ Trucks, b$\longrightarrow$ VehicleCargoCarrier
Factory $\longrightarrow$ **VehicleCargoCarrier** :
H4; a$\longrightarrow$ Vehicle,b$\longrightarrow$ CargoCarrier, c$\longrightarrow$ VehicleCargoCarrier

Table 1: Specification of the articulation rules shown in Figure 3

2.3 Integration graphs: Formal definition

A (di)graph is a pair $G = (N, E)$ where N is a finite nonempty set of *nodes* and E is a multiset on $N \times N$. If $e = (n_1, n_2) \in E$ is a (directed) edge we'll write simply $e = n_1 n_2$. The basic objects of our model are called *source graphs*. A source graph is a pair $SG = (G, VN)$, where G is a graph and VN is a subset of the nodes of G called the set of visible nodes of G. A source graph describes a world by means of the graph G and offers an integration interface with other worlds by means of the set of its visible nodes VN. Using the graphs in the Figure 5, $SG1 = (G1, a, b, c, d)$ and $SG2 = (G3, m, n, p)$are examples of source graphs.

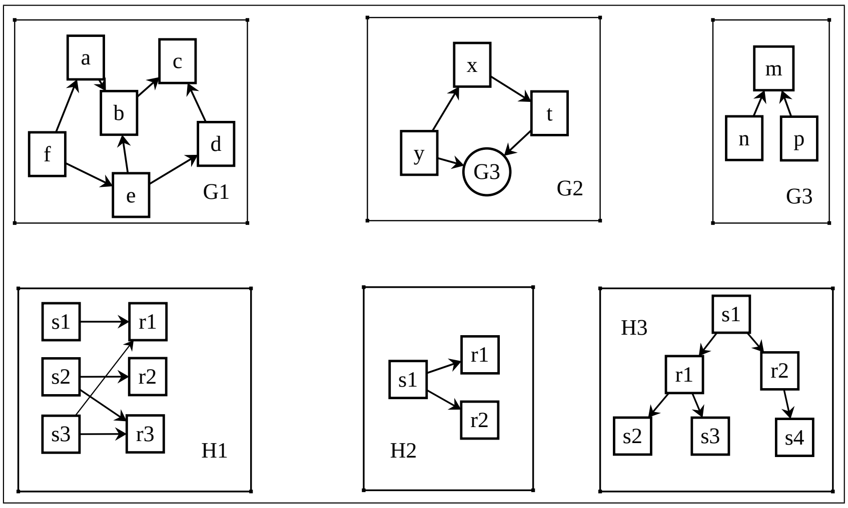

Figure 5: Composition patterns

In order to describe the interaction between different worlds (i. e. sources graphs) we define another basic object *the articulation rule*. An articulation rule is a bipartite (directed) graph $H = (S, R; E)$ having the set of nodes $S \cup R$ (where S and R are disjoint, nonempty sets) and E a set of directed edges joining nodes from S to nodes from R. In the Figure 4, three articulation rules $H1, H2$ and $H3$ are described.

An integration graph is a pair $IG = (G, VN)$, where G is a graph and VN is a set of visible nodes. Each node $n \in N(G)$ of G is either a *simple node* or a *complex node*. The set of visible nodes associated to a simple node n is the singleton $\{n\}$.

If n is a complex node, then it has associated a description $D(n)$ which is also an integration graph. If $e = n_1n_2 \in E(G)$ is and edge of G, and if at least one of its extremities is complex node, then the edge e has associated an articulation rule H_e and two injective functions s_e and r_e. The function r_e associates to each node of R_{H_e} a node from the visible set of nodes of n_1. The function s_e associates to each node of S_{H_e} a node from the visible set of nodes of n_2. The triple (H_e, s_e, r_e) means that the links between the two nodes are described by the pattern $H(e)$ and can be obtained by using the two substitutions s_e and r_e.

An example of integration graph is depicted in Figure 6, where the rectangle node labelled Art is a simple node and the elliptical nodes labelled $G1$ and $G2$ are complex nodes that refer to the corresponding graphs. The flat graph that can be obtained from the integration graph after applying all our articulation rules and by only considering the visible specified nodes is illustrated in Figure 7.

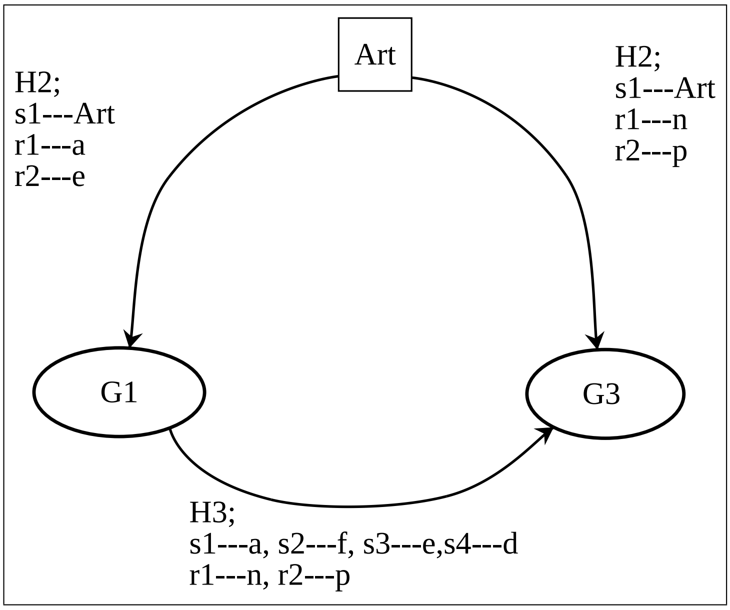

Figure 6: Example of integration graph

3 Discussion

Graph-based models such as semantic network and conceptual graphs could be potentially effective to express mappings between concepts because of their overloaded constructors (e.g. nodes or edges). For instance, these could easily allow the mapping of properties into concepts

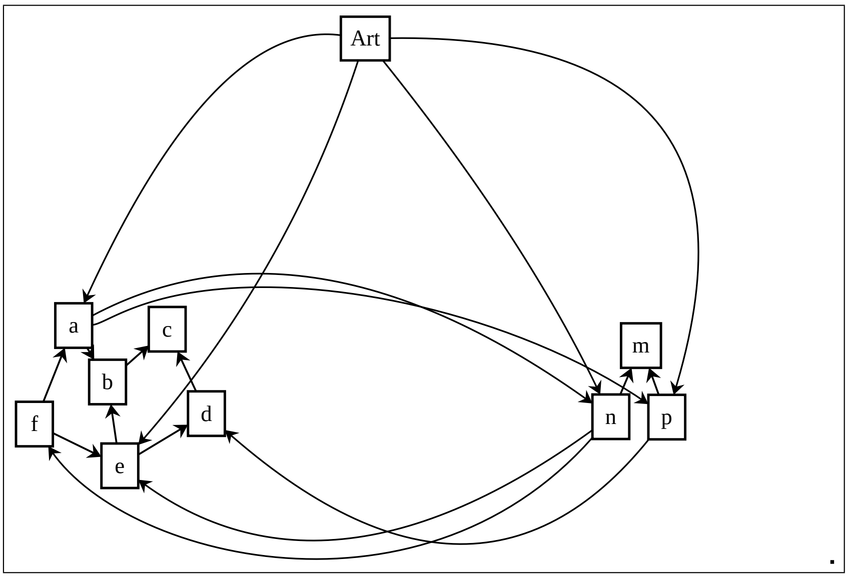

Figure 7: The flat graph represented in Figure 6

and vice versa. Conversely, this kind of mapping cannot be naturally and easily introduced in frame-based models (including DL-based ones), because of their structural rigidity. However, the expressiveness of standard graph-based approach is not adequate to model the DKBs to be articulated, integrated or merged. Moreover, the absence of a full mapping into a formal logical system may prevent non-trivial automate deductions. On the other side, frame-based approaches support hierarchical reasoning (e.g. subsumption) in a straightforward way.

In order to overcome the drawbacks of both knowledge models, we have introduced integration graphs (i.e. a type of hierarchical graphs) that combine the flexibility of traditional graph models with the hierarchical structure of frame models. being based on FOPL, hierarchical graphs allow the introduction of automated semantic reasoning services based on their knowledge model. Our integration approach can easily capture simple semantic implications rules or simple Horn clauses such as the ones introduced in [9], which are necessary for domain knowledge interoperability. We note that our approach assumes that a frame-based knowledge model is used to describe the DKBs to be articulatedm integrated or unified, while integration graphs are used to describe the articulation of these DKBs.

An important thing about complex nodes in our integration graphs is their interface. This is an interesting new tool for both taking induced subgraphs in the hierarchical structure and also to describe links necessary for navigation in the nested structured worlds. Our principal aim in considering such graphs was to have a formalism useful not only in describing the relationships between data but also for capture the need to navigate through paths in the (semi)structured worlds of data. Of course, low level operations such adding or deleting simple or complex nodes can be easily designed. If the source graphs of the integration graph are chosen to be conceptual graphs, the entire structure will provide an elegant mode to do fusion reasoning by a method similar to one used in [4] for nested graphs.

The formalism described in this paper will be incorporated in the ongoing information integration project Infoshare, a formal approach for sharing the information contained in two or more relational or object-oriented databases using the articulation of their conceptual schemas. The objectives of the InfoShaRe project include, among others, the possibility of reusing views and queries from a shared repository in order to support the articulation process in a fully automated way.

References

[1] F. Baader et al., editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] D. Beneventano et al. The MOMIS approach to Information Integration. In *IEEE and AAAI Intl. Conf. on Enterprise Information Systems (ICEIS'01)*, 2001.

[3] G. Busatto. *An Abstract Model of Hierarchical Graphs and Hierarchical Graph Transformation*. PhD thesis, Paderborn University, Germany, 2002.

[4] M. Chein, M.-L. Mugnier, and G. Simonet. A graph-based knowledge representation model with FOL semantics. In *Proc. of the 6th Intl. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 524–534, 1998.

[5] E. Compatangelo and H. Meisel. EER-ConceptTool: a "reasonable" environment for schema and ontology sharing. In *Proc. of the 14th IEEE Intl. Conf. on Tools with Artificial Intelligence (ICTAI'2002)*, pages 527–534, 2002.

[6] P. D. Karp. The Design Space of Frame Knowledge Representation Systems. Technical Note 520, Artificial Intelligence Center, SRI International, 1993.

[7] A. Maedcke et al. MAFRA - A MApping FRAmework for Distributed Ontologies. In *Proc. of the 13th Intl. Conf. on Knowledge Engineering and Knowledge Management (EKAW'2002)*, volume 2473 of *Lecture Notes in Computer Science*, pages 235–250. Springer-Verlag, 2002.

[8] P. Mitra, G. Wiederhold, and J. Jannink. Semi-automatic integration of knowledge sources. In *Proc. of the 2nd Intl. Conf. On Information Fusion (FUSION'99)*, 1999.

[9] P. Mitra, G. Wiederhold, and M. L. Kersten. A Graph-Oriented Model for the Articulation of Ontology Interdependencies. In *Proc. of the VII Conf. on Extending Database Technology (EDBT'2000)*, Lecture Notes in Computer Science, pages 86–100. Springer-Verlag, 2000.

[10] J. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., 2000.

[11] M. Uschold. Knowledge level modelling: concepts and terminology. *The Knowledge Engineering Review*, 13(1):5–29, 1998.