

# Interactive Schema Matching With Semantic Functions

Guilian Wang<sup>1</sup> Joseph Goguen<sup>1</sup> Young-Kwang Nam<sup>2</sup> Kai Lin<sup>3</sup>

<sup>1</sup>Department of Computer Science and Engineering, U.C. San Diego. {guilian, goguen}@cs.ucsd.edu

<sup>2</sup>Department of Computer Science, Yonsei University, Korea. yknam@dragon.yonsei.ac.kr

<sup>3</sup>San Diego Supercomputer Center, U.C. San Diego. klin@sdsc.edu

## Abstract

Experience suggests that fully automated schema matching is infeasible, especially when n-to-m matches involving semantic functions are required. It is therefore advisable for a matching algorithm not only to do as much as possible automatically, but also to accurately identify positions where user input is necessary. Our matching algorithm combines several existing approaches with a new emphasis on using the context provided by the way elements are embedded in paths. A prototype tested on biological data (gene sequence, DNA, RNA, etc.) and on bibliographic data shows significant performance improvements, and will soon be included in our DDXMI data integration system. Some ideas for further improving the algorithm are also discussed.

## 1. Introduction

Many emerging scientific and commercial applications use multiple distributed information sources, having various forms of metadata, of which schema are the most important. Discovering semantic correspondences between different schema is therefore a critical step in many such applications: a) for data warehousing, to help find simple, correct data transformations from the source schemas to a single target schema; b) for virtual data integration, to provide basis for rewriting a user query over a mediated schema to optimized sub-queries over the relevant source schemas (this often called query discovery) [ED01, LC00]; c) for schema integration, to find similar structures (or "integration points") across multiple schemas [BCV99]; d) for e-business or scientific workflow, to identify semantically correct mappings of messages, often in XML format, from one service or step to the next.

The schema matching problem takes as input two schemas and possibly some auxiliary information, and returns a mapping that identifies elements that correspond semantically [DR02, MBR01, RB01]. This can be very challenging: even schemas for the same entities from different sources may have very different structural and naming conventions, and may also use different data models. Similar, or even the same, labels may be used for schema elements having totally different meanings, or having subtly different semantics, due to differences in the unit, precision, resolution, aggregation, measurement protocol, etc.; this is extremely common in domains like environmental science, ecology, biology, chemistry and physics.

In the bibliographic domain, in addition to naming and structural differences, different units are used for price of books, some may not give the publisher (e.g., when the data source provider is the publisher for all the books described by that schema), price or publication date may be missing, some may use the full name of authors, while others use separate first and last names, and some may have different scope from others, e.g. including articles as well as books.

An extensive review of existing techniques and tools for automatic schema mapping is given in [RB01], showing that existing tools only help find 1:1 matches, and have great difficulty with complex matches that involve conditions, or transformation or conversion functions; furthermore, n:m relations for n or m > 1, cannot be discovered automatically at all. In practice, schema mapping is done manually by domain experts, usually with a graphical tool [MHH00, NGW02], and is very time consuming when there are many data sources or when schemas are very large and/or complex.

Our goal is to automatically discover simple correspondences as effectively as possible by combining existing schema mapping techniques, and prompting the user to provide specific input at critical points when no adequate 1:1 match can be found; these points are where there are more than one almost equally good match, and also where n:m correspondences and/or functions may be needed. Another contribution is to utilize the context information provided by the paths of schema elements. A research prototype is partially implemented and preliminarily tests have been run using bibliographic and biological data sets, with encouraging results.

## 2. Related work

A recent survey of automatic schema matching [RB01] classifies tools by how they use similarity of various aspects of the information that schemas carry (i.e. element name, structure, data type, constraints), data content, and auxiliary information (i.e. general domain specific common terminologies), either by using one technique or

combining several. LSD and GLUE use machine learning to evaluate data instances and train matchers, and then predict element similarities by combining their results [DDH01, Doan03]. Cupid combines name and structural matching, and predicts element similarity based on the similarity of the name and data type of their components (sub-elements and attributes in XML DTD or Schema), which can easily lead to incorrectly identifying atomic elements (leaves in schema tree). SF and Rondo provide a versatile graph matching algorithm that is also used in our prototype for structural matching, and also provide more realistic metrics for match accuracy, which are used in COMA and our tool. COMA supports flexibly combining different schema matching techniques, emphasizes reuse of previous matching results, and allows user input during matching; it has been carefully tested for real purchase orders schemas, showing that reuse of existing mappings can help a lot. However, no existing work considers the n:m correspondences and semantic functions that are needed in many practical applications.

### 3. Our approach

The most novel aspects of our approach are its support for n:m matches that involve semantic functions, and its focus on providing maximum support to users, rather than total automation. It enhances, combines, and reuses algorithms for linguistic, structural, and graph matching from prior work. Its structural matching considers closely connected nodes, as in ARTEMIS, COMA, Cupid, DIKE, and Rondo. It has a series of matching stages using different methods, as in COMA and Cupid; results from different methods are combined by weighted summation. Linguistic similarity of elements is based not just on tag strings, but also on their path strings, as in COMA, so that hierarchical schema structure is treated as linguistic, and paths provide context dependent matches for shared elements, as in Cupid. The following are some more technical characteristics that differ from prior work:

a. Schemas are represented in both tree and graph formats: the tree format is convenient for context and sub-tree matching, while the graph format accommodates additional schema information, including element type (attribute, atomic, or context), data type, and constraint.

b. Specific user input is interactively requested at critical points, not just at pre- and/or post-match, and user supplied matches are not modified later; this makes post-match editing much easier, since bad guesses made without user guidance need not be corrected and do not propagate.

c. Graph matching gives new matches based on user supplied and/or high similarity value matches from previous stages, without changing these "good matches" in later stages.

d. Noting that tag meanings can vary with context, and that context is given by preceding (i.e., higher) elements, our approach seeks to identify "core" contexts and attributes, which are the most important contextualizing elements for tags within subtrees, by using heuristics and user input; then threshold and context are checked for them. For example, in matching two book DTDs, /arts/book and /arts/article are found to be core contexts, with title, author, publisher as their main attributes. Previous steps found that the best match with /arts/article is /bookstore/book, but its similarity value that is lower than the core threshold, so that matches in its subtree are not reliable, and user input is needed. Matching /arts/book/publisher/name with /bookstore/book/author/name fails because they are in different contexts, even though they have the same tag.

e. When refining subtree matches in context checking, if the roots of 2 sub-trees are considered a "good" match, then the path of an element in the subtree may be changed into a shorter one starting at the corresponding subtree root instead of the root of the whole tree. This helps to reduce the impact of heterogeneous structure on path similarity of elements in the sub-trees. For example, in matching two bio-molecular DTDs, after finding that /Bsm1/Definitions/Sequences/Sequence/Feature-tables/Feature-table/Feature matches /Sequences/Sequence /Segment, then for all nodes in the subtree of Feature, the string before /Feature in its path is cut off, and for all the nodes in subtree of Segment, the string before /Segment in its path is also cut off. This should yield to higher path similarity for (Feature/id, Segment/seg-id), (Feature/Location/Interval-loc/startpos, Segment/seg-start), etc.

f. Pre-order traversal gives top-down matching, which is more efficient than bottom-up, and our combination of techniques, earlier identification of core contexts, and user input, all help to improve performance on schemas that differ at higher levels.

The approach has the following steps:

1) **Preprocessing**: parsing two schema files, for target and for source, and producing a tree and a graph representation for each.

**Tree representation**: Each element and attribute of a schema is a node in the schema tree and sub-elements are child nodes. Each node has its type (attribute, leaf, or parent that is possibly a context), data type, height, a set of matches  $(n_j, sim)$  for node  $n_j$  in the other tree, constraints, etc.

**Graph representation**: The RDF model (directed labeled graph) is used. Each element and attribute that is a node,  $n_i$ , in the above tree representation, becomes an RDF resource,  $res_{n_i}$ . Each node type  $nt_j$  and data type  $dt_k$

become RDF resources  $res_{n_i}$  and  $res_{n_j}$  respectively. Corresponding edges are created for each node and added to the graph, two for its parent if it is not the root node,  $(res_{n_i}, parent\_height_{n_i}, parent_{n_i})$  and  $(parent_{n_i}, child\_height_{n_i}, res_{n_i})$ , one for its node type,  $(res_{n_i}, nodeType, res_{n_i})$ , one for its data type:  $(res_{n_i}, dataType, res_{d_{mi}})$ , etc.

2) **Linguistic matching**: In the current prototype, linguistic matching includes a) terminological matching, in which element name and path are normalized through domain specific thesauri containing a synonym table, an abbreviation table and a hyponym table; and b) syntactic matching, which is purely based on string similarity, computed according to the number of substring they have in common. The implementation in [MRB03] is borrowed in our current prototype. For any pair of two nodes  $(n_i, n_j)$  in the two trees, both normalized name similarity value  $ns_{i,j}$  and normalized path similarity value  $ps_{i,j}$  are computed.

$$ns_{i,j} = Coeff * StringMatch(Normalize(name_{n_i}, name_{n_j})),$$

$$ps_{i,j} = Coeff * StringMatch(Normalize(path_{n_i}, path_{n_j})), \text{ where } Coeff = 0.8 \text{ if one of } name_{n_i} \text{ and } name_{n_j} \text{ happen to be hyponym to another, otherwise } Coeff = 1.0.$$

The linguistic similarity value,  $ls_{i,j}$ , is the weighted summation of name and path similarities

$$ls_{i,j} = ns_{i,j} * w_{n_i} + ps_{i,j} * w_{p_i}, \text{ where } w_{n_i} + w_{p_i} = 1.0$$

3) **Structural matching**: the Similarity Flooding algorithm for graph matching in [MMR02] is extended to be the structural matcher of our current prototype. Taking as input graphs  $G, G_2$ , and a set of initial similarity values,  $InitMap$ , between the nodes of the graphs, it iteratively propagates the initial similarity of nodes to the surrounding nodes, using the intuition that neighbors of similar nodes are similar. The original algorithm is modified to keep input mappings that are from user and/or of high similarity from previous steps for some nodes unaffected. The initial mappings,  $InitMap$ , are selected from the best result for each target node from the linguistic matching step by filtering out those best mappings whose  $max\_ls_i = w_{threshold\_g}$ .

$$gs_{i,j} = GraphMatch(G, G_s, InitMap)$$

The following shows how it works. The `s_bib.dtd` and `s_arts.dtd` in Figure 1 and Figure 2 are simplified to provide more readable graphs. Their graph representations are shown in Figures 3 and 4 (The edge label `cl1` stands for “child at level 1”, `dt` for “data type”)

```
<!ELEMENT bib (book*)>
<!ELEMENT book (title)>
<!ATTLIST book year CDATA #REQUIRED >
<!ELEMENT title CDATA>
```

Figure 1. `s_bib.dtd`

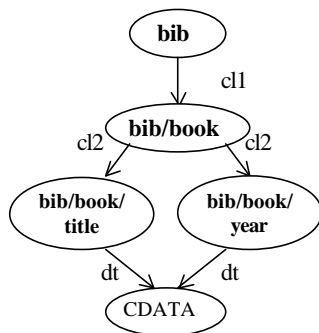


Figure 3. The graph for `s_bib.dtd`

```
<!ELEMENT arts (book | article) >
<!ELEMENT book (title)>
<!ATTLIST book date CDATA #REQUIRED >
<!ELEMENT title CDATA>
<!ELEMENT article (title)>
<!ATTLIST article date CDATA #REQUIRED >
```

Figure 2. `s_arts.dtd`

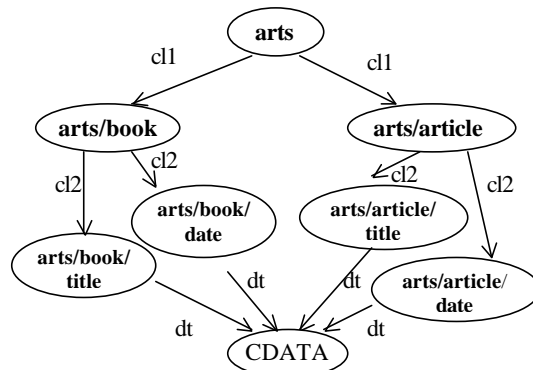


Figure 4. The graph for `s_arts.dtd`

The  $InitMap$  contains  $(bib/book/title, arts/book/title, 0.77)$  from the linguistic matching step, and built-in matches for DTD node and data types, such as  $(parentElem, parentElem, 1.0)$ ,  $(leafElem, leafElem, 1.0)$ ,  $(attribute, attribute, 1.0)$ ,  $(PCDATA, PCDATA, 1.0)$ ,  $(CDATA, CDATA, 1.0)$ .

The pairwise connectivity graph is shown in Figure 5. The positive value on top of a mapping pair node is its initial similarity, or else is zero if not shown. The propagation graph in Figure 6 is induced from this pairwise connectivity graph (note: the added propagation edges of the 6 nodes at lower part are not shown in order to reduce the complexity). For each edge  $(source, label, target)$ , an opposite edge is added for propagating  $target$  node's similarity to the  $source$  node. The edge  $label$  is changed to a  $weight$  value that means how much of the similarity value of the source node will propagate to the target node in each iteration of propagation. The  $weight$  value is

determined by the number of outgoing same *label* edges from the *source* node, and whether the different target nodes are treated equally or weighted. Here we treat all targets equally, so the weight for each outgoing same label edge is 1.0 divided by the number of these edges. Several fixpoint formula were explored in [MMR02], the basic one is that after each iteration, the map pair node similarity  $sim^{n+1} = \text{Normalize}(sim^n + \mathbf{S} \text{ weight} \times sim_{source}$  for each incoming edge). Propagation continues until the Euclidean length of the residual vector  $\Delta(sim^{n+1}, sim^n)$  is less than a threshold. Our modifications to the original algorithm: a) “good” map pair nodes propagate their similarity to neighbors, but neighbors don’t propagate to them, which means that the propagation formula are applied only to “non-good” map pair nodes; and b) “good” matching nodes don’t join the normalization process.

In Figure 6, the final similarity values ( $sim1, sim2$ ) are shown around each map pair node, where  $sim1$  is from the original algorithm without separating shared elements in the input graph, and  $sim2$  is from the modified algorithm that fixes high similarity initial mappings and uses paths instead of elements to make shared elements context-dependent. The second gives higher similarity values for correct matches and lower similarity values for most wrong matches, although it doesn’t differentiate (bib/book, arts/book, 0.88) from (bib/book, arts/article, 0.88), the later combination of linguistic and structural similarity makes (bib/book, arts/book) win significantly. The first method doesn’t differentiate (year, title, 0.42) from (year, date, 0.42) since they have the same similarity value, and gives with too low similarity for (bib, arts, 0.29), both of which cause difficulty on filtering.

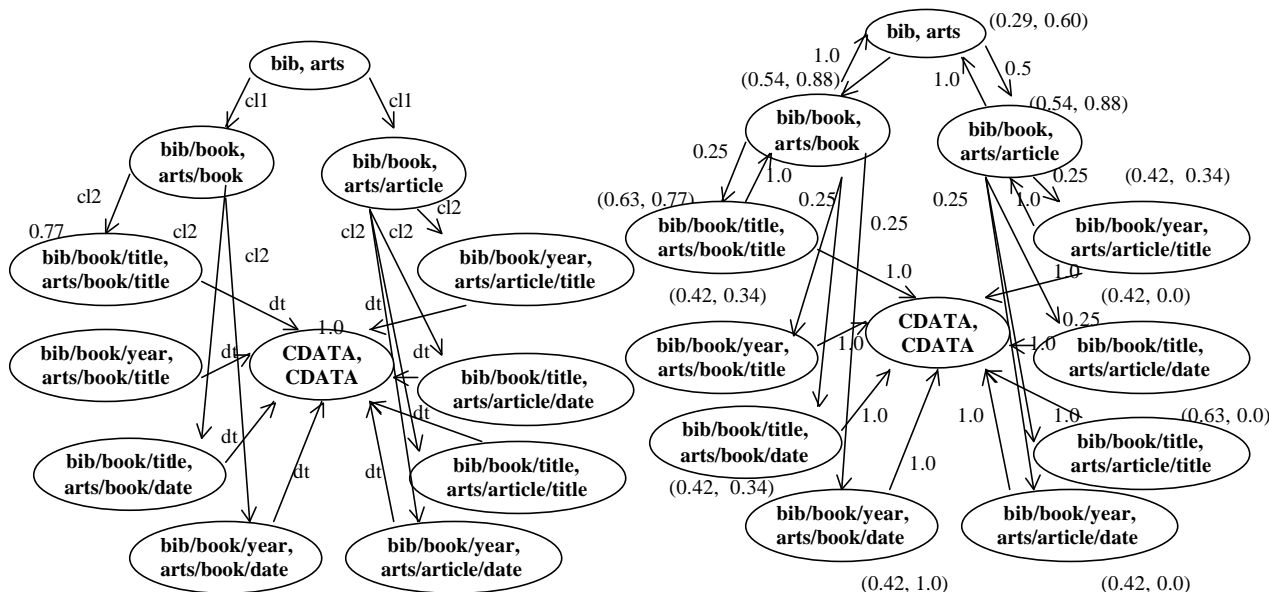


Figure 5. Pairwise connectivity graph

Figure 6. Induced propagation graph from Figure 5

After this graph matching step, the similarity between two tree nodes is the weighted sum of similarity values of name, path and structure:

$$s_{ij} = ns_{ij} \times w_{n_c} + ps_{ij} \times w_{p_c} + gs_{ij} \times w_{g_c}, \text{ where } w_{n_l} + w_{p_l} + w_{g_c} = 1.0 \quad (a)$$

4) **Context consistency enforcement step:** Mapping is refined by checking and (possibly multiple) re-matching sub-trees rooted at the core elements. See the discussion in 3d above.

After the core contexts have been identified, they are sorted in ascending order according to their height from the tree root in order to do top-down matching refinement. For each node  $n_i$ , check whether its best matching node that has the maximum similarity bigger than a relatively high similarity threshold. If not, for this important target node, there may be no good match from the combined linguistic and structural matching steps, so that its sub-tree matches might also not be good, and user interaction is requested, or sub-tree matches are thrown away in the non-interactive mode. Otherwise, refinement of subtree matches to best matching node  $n_j$ 's subtree starts with initial mappings,  $InitMap_{subtree_{ni}, subtree_{nj}}$  which are the best so far within these two sub-trees (not the best at whole trees instead) and with similarity bigger than a threshold, subtree graphs  $G_{subtree_{ni}}$  and  $G_{subtree_{nj}}$  are generated as input

with  $InitMap_{subtree_{ni}, subtree_{nj}}$  for  $GraphMatch$  to compute again the structural similarity of nodes  $n_h$  in  $n_i$ 's sub-tree and nodes  $n_k$  in  $n_j$ 's sub-tree. Then the similarity index between  $n_h$  and  $n_k$  is recomputed using formula (a).

$$g_{s_h, k} = GraphMatch(G_{subtree_{ni}}, G_{subtree_{nj}}, InitMap_{subtree_{ni}, subtree_{nj}})$$

#### 4. Examples and results

Two application domains have been used to evaluate our approach: 1) three book DTDs obtained from the web, bibliography (*b*), arts (*a*), bookstore (*st*), and a mediated one (*bk*), have been tested in our data integration prototype system DDXMI, using manual matching [NGW02]; and 2) three bio-molecular DTDs, GAME (*g*), BSML (*bs*), BIOML (*bi*), which have been trimmed to remove some branches that are huge but irrelevant to sequence encoding, or were caused by unsophisticated design, plus a mediated DTD (*s*) for gene sequence encoding only. The characteristics of relevant schemas are shown in Table 1.

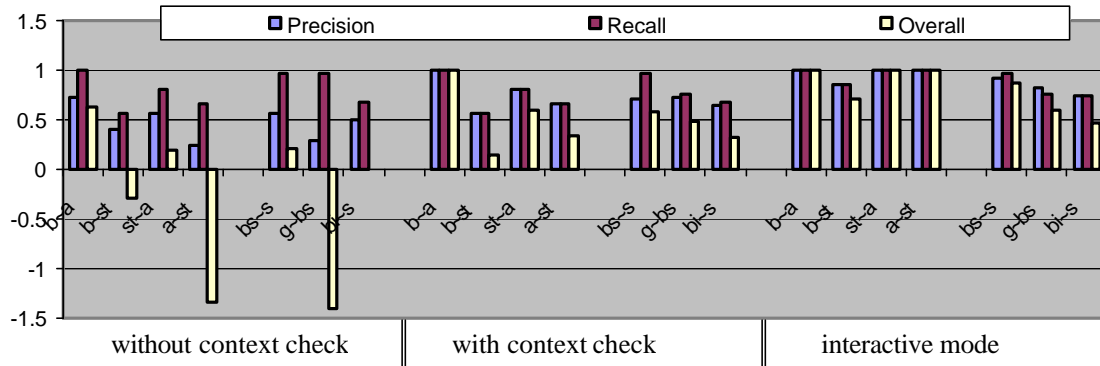
Schema	#Path	#Nodes	Max Depth	Schema	#Path	#Nodes	Max Depth
<i>b</i>	13	11	4	<i>bs</i>	69	58	10
<i>st</i>	7	7	4	<i>g</i>	124	66	7
<i>a</i>	21	15	5	<i>bi</i>	32	21	8
<i>bk</i>	13	13	5	<i>s</i>	30	30	4

**Table 1.** Characteristics of schemas involved in the tested schema match tasks

To evaluate the quality of the mappings from our matching process, we use the same measures used in [DR02, MMR02], derived from the information retrieval and data mining fields. The manually determined real matches ( $R$ ) for a match task are compared with the matches,  $P$ , returned by automatic matching process. The following are counted: the correctly identified matches  $T$ , the wrong matches  $F = P - T$ , and the missed matches  $M = R - T$ . These correspond to true positives, false positives, and false negatives respectively. Then the following three measures are computed:

- $Precision = T/P = T/(T+F)$
- $Recall = T/R$
- $Overall = 1 - (F+M)/R = (I-F)/R = Recall * (2 - 1/Precision)$

The first gives the reliability of the mapping, the second gives the share of real matches found, and the third is a combined measure for mapping quality, taking account of the post-match effort needed for both removing wrong and adding missed matches.



**Figure 7.** Measure values of tested match tasks

#### 5. Summary and future work

Test results in Figure 7 show that our approach works well, in comparison with the purchase order schema tests of COMA [DR02] and Cupid [MBR01]. Our context consistency step seems especially helpful, improving the overall score significantly. And user interactions help identify n:m matches with semantic functions where good 1:1 matches can not be found during matching process. Some of the overall measures are still very low because the difference of price units for book DTDs can't be detected without analyzing the data content or having more

semantic metadata, or because necessary conditions can't be found, or even detected that are needed for matching according to the available information.

In addition, many issues remain to be studied. The first is representing the generated mappings to facilitate user editing, for correcting false matches, adding missed matches, attaching restructuring and semantic conversion functions, making sure that all heterogeneities are resolved for the next step, query discovery, which is important for data integration. The second issue is finding n:m matches that involve semantic functions; these are complex, but common in practice. The third issue is semantic metadata, which could be made semantic matching more convenient and reliable, for example, to help with the second issue; we are now studying this problem for ecological data integration and analysis problems. The fourth issue is using ontologies to help with schema mapping, assuming an ontology is attached to each schema; we are exploring ontology mappings and developing support techniques [NM00]. Match composition and reuse also deserve further research for the above issues, and have been touched upon for the 1:1 case by [DR02, MB02]. Finally, we have not yet provided the interface for delivering user input on detected mismatches, although this should be done soon.

### Acknowledgements

We thank Bertram Ludäscher for his invaluable discussions. This material is based upon work supported by the National Science Foundation under Grant No. 0225676.

### References

- [BCV99] Bergamaschi, S., S. Castano, and M. Vincini: Semantic Integration of Semistructured and Structured Data Sources. *SIGMOD Record* 28(1), 1999, 54-59.
- [DDH01] Doan, A, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD '01*, 2001.
- [DR02] Do, H. and E. Rahm. Coma - A System for flexible combination of schema matching approaches. In *Proceedings of the 28th Conf. on Very Large Databases (VLDB)*, 2002.
- [Doan03] Doan, A. Thesis. <http://anhai.cs.uiuc.edu/home/thesis/anhai-thesis.pdf>
- [MBR01] Madhavan, J., P. A. Bernstein, E. Rahm. Generic Schema Matching with Cupid, *Proc. 27th Int. Conf. on Very Large Data Bases (VLDB)*, 2001.
- [ED01] Embley, D.W. et al. Multifaceted Exploitation of Metadata for attribute Match Discovery in Information Integration. *WIIW 2001*.
- [LC00] Li, W., C. Clifton. SemInt: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural network. *Data and Knowledge Engineering* 33:1, p49-84, 2000.
- [MB02] Madhavan, J., P. A. Bernstein, P. Domingos, & A.Y. Halevy. Representing and Reasoning About Mappings between Domain Models. In *18th National Conference on Artificial Intelligence (AAAI)*, 2002.
- [MRB03] Melnik, S., E. Rahm, P. A. Bernstein. Rondo: A Programming Platform for Generic Model Management. In *ACM Intl. Conference on Management of Data (SIGMOD)*, San Diego, CA, 2003.
- [MZ98] Milo, T. and S. Zohar: Using Schema Matching to Simplify Heterogeneous Data Translation. *VLDB 1998*, 122-133.
- [MHH00] Miller, R., L. Haas, and M.A. Hernandez: Schema Mapping as Query Discovery. *VLDB 2000*, 77-88.
- [MMR02] Melnik, S., H. Garcia-Molina, E. Rahm: Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. *ICDE 2002*.
- [NGW02] Nam, Y. K, J. Goguen, and G. Wang. A Metadata Integration Assistant Generator for Heterogeneous Distributed Databases. In *Proceedings of International Conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems*, Springer, Lecture Notes in Computer Science, Volume 2519, 2002, pages 1332-1344.
- [NM00] Noy, N. F. and M. A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment *In the Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX. Available as SMI technical report [SMI-2000-0831](http://www.smi.stg.cmu.edu/SMI-2000-0831) (2000).
- [RB01] Rahm, E. and Bernstein, P. A.. On Matching Schemas Automatically, *Techn. Report 1/2001*, Dept. of Comp. Science, Univ. of Leipzig. <http://dol.uni-leipzig.de/pub/2001-5/en>