Evaluating Matching Algorithms: the Monotonicity Principle
Position Statement

Avigdor Gal
Technion – Israel Institute of Technology
avigal@ie.technion.ac.il

Traditionally, semantic reconciliation was performed by a human observer (a designer or a DBA) [8] due to its complexity [3]. However, manual reconciliation (with or without computer-aided tools) tends to be slow and inefficient in dynamic environments and does not scale for obvious reasons. Therefore, the introduction of the semantic Web vision and the shift towards machine understandable Web resources has unearthed the importance of automatic semantic reconciliation. Consequently, new tools for automating the process, such as GLUE [4], and OntoBuilder [11], were introduced.

Generally speaking, the process of semantic reconciliation is performed in two steps. First, given two attribute sets $\mathcal{A}$ and $\mathcal{A}'$ (denoted *schemata*) with n_1 and n_2 attributes, respectively,[1] a degree of similarity is computed **automatically** for all attribute pairs (one attribute from each schema),[2] using such methods as name matching, domain matching, structure (such as XML hierarchical representation) matching, and Machine Learning techniques. As a second step, a single mapping from $\mathcal{A}$ to $\mathcal{A}'$ is chosen to be the *best mapping*. Typically, the best mapping is the one that maximizes the sum (or average) of pair-wise weights of the selected attributes. We differentiate the best mapping from the *exact mapping*, which is the output of a matching process as would be performed by a human observer.

Automatic matching may carry with it a degree of uncertainty since "the syntactic representation of schemas and data do not completely convey the semantics of different databases" [10]. As an example, consider name matching, a common method in tools such as OntoBuilder [6], Protégé [5], and Ariadne [9]. With name matching, one assumes that similar attributes have similar (or even identical) names. However, the occurrence of synonyms (*e.g.*, `remuneration` and `salary`) and homonyms (*e.g.*, `age` referring to either human age or wine age) may trap this method into erroneous mapping. As a consequence, there is no guarantee that the exact mapping is always the best mapping.

We present the *monotonicity principle*, a sufficient condition to ensure that exact mapping would be ranked sufficiently close to the best mapping. Roughly speaking, the monotonicity principle proclaims that by replacing a mapping with a better one, score wise, one gets a more accurate mapping (from a human observer point of view), even if by doing so, some of the attribute mappings are of less quality. We have demonstrated, through theoretical [7] and empirical analysis,[2] that for monotonic mappings that satisfy the monotonicity principle, one can safely interpret a high similarity measure as an indication that more attributes are mapped correctly. An immediate consequence of this result is the establishment of a corroboration for the quality of mapping algorithms, based on their capability to generate monotonic mappings. We have experimented with a matching algorithm and report on our experiences in [2]. Our findings indicate that matching algorithms that generate monotonic mappings are well-suited for automatic semantic reconciliation. Another outcome of the monotonicity principle is that a good automatic semantic reconciliation algorithm would rank the exact mapping relatively close to the best mapping, thus enabling an efficient search of the exact mapping [1].

Monotonicity is not defined in "operational" terms, since it is compared to an initially unknown exact mapping. In fact, such an operational definition may not be generally developed, since algorithms may perform well only on some schema pairs. Therefore, a task for future research involves possible classification of application types on which

[1]The use of relational terms is in no way restrictive, and is used here to avoid the introduction of an extensive terminology that is of little benefit in this paper.

[2]Extensions to this basic model (*e.g.*, [10]) are beyond the scope of this statement.

certain algorithms would work better than others. Best mappings may also be subjective at times (less so in the type of applications we were exploring, though). It is not clear at this time how an operational definition can be developed in such cases without personalizing the algorithms to specific human observers. Taken to the extreme, an adaptive algorithm would rank erroneous mappings higher, simply by following a human observer presumptions. This line of research is also left for future investigation.

The recent steps taken in the direction of automating semantic reconciliation highlight the critical need of this research. As the automation of the process has already begun to take shape, often without the benefits of thorough research, the study is timely. We envision multitude of applications of automatic schema matching to the semantic Web. For example, we are currently designing smart agents that negotiate over information goods using schema information and can combat schema heterogeneity.

Acknowledgments

The work was partially supported by Technion V.P.R. Fund - New York Metropolitan Research Fund, the Ministry of Science, Culture, and Sport in Israel and by the CNR in Italy, and the IBM Faculty Award for 2002/2003 on ”Self-Configuration in Autonomic Computing using Knowledge Management.”

References

[1] A. Anaby-Tavor, A. Gal, and A. Moss. Efficient algorithms for top-k matchings. Submitted for publication. Available upon request from avigal@ie.technion.ac.il, 2003.

[2] A. Anaby-Tavor, A. Gal, and A. Trombetta. Evaluating matching algorithms: the monotonicity principle. In S. Kambhampati and Craig A. Knoblock, editors, *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web*, pages 47–52, Acapulco, Mexico, August 2003.

[3] B. Convent. Unsolvable problems related to the view integration approach. In *Proceedings of the International Conference on Database Theory (ICDT)*, Rome, Italy, September 1986. In *Computer Science*' Vol. 243, G. Goos and J. Hartmanis, Eds. Springer-Verlag, New York, pp. 141-156.

[4] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the eleventh international conference on World Wide Web*, pages 662–673. ACM Press, 2002.

[5] N. Fridman Noy and M.A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 450–455, Austin, TX, 2000.

[6] A. Gal, G. Modica, and H.M. Jamil. Improving web search with automatic ontology matching. Submitted for publication. Available upon request from avigal@ie.technion.ac.il, 2003.

[7] A. Gal, A. Trombetta, A. Anaby-Tavor, and D. Montesi. A model for schema integration in heterogeneous databases. In *Proceedings of the 7th International Database Engineering and Application Symposium*, Hong Kong, China, July 2003.

[8] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 51–61. ACM Press, 1997.

[9] C.A. Knoblock, S. Minton, J.L. Ambite, N. Ashish, I. Muslea, A. Philpot, and S. Tejada. The Ariadne approach to web-based information integration. *International Journal of Cooperative Information Systems (IJCIS)*, 10(1-2):145–169, 2001.

[10] R.J. Miller, L.M. Haas, and M.A. Hernández. Schema mapping as query discovery. In A. El Abbadi, M.L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *Proceedings of the International conference on very Large Data Bases (VLDB)*, pages 77–88. Morgan Kaufmann, 2000.

[11] G. Modica, A. Gal, and H. Jamil. The use of machine-generated ontologies in dynamic information seeking. In C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, editors, *Cooperative Information Systems, 9th International Conference, CoopIS 2001, Trento, Italy, September 5-7, 2001, Proceedings*, volume 2172 of *Lecture Notes in Computer Science*, pages 433–448. Springer, 2001.