MIT Sloan School of Management

Working Paper 4345-02
February 2002

AUTOMATED NEGOTIATION FROM DECLARATIVE CONTRACT DESCRIPTIONS

Ashish Mishra, Michael Ripley, Amar Gupta

Using Structural Analysis to Mediate XML Semantic Interoperability

Ashish Mishra, Michael Ripley, Amar Gupta
Email: ashm@mit.edu, rip@mitre.org, agupta@mit.edu

Abstract

At the forefront of interoperability using XML in an Internet environment is the issue of semantic translation; that is, the ability to properly interpret the elements, attributes, and values contained in an XML file. In many cases, specific domains have standardized the way data are represented in XML. When this does not occur, some type of mediation is required to interpret XML formatted data that does not adhere to pre-defined semantics. The prototype X-Map was developed to investigate what is required to mediate semantic interoperability between heterogeneous domains. An essential component of this system is structural analysis of data representations in the respective domains. When mediating XML data between similar but non-identical domains, we cannot rely solely on semantic similarities of tags and/or the data content of elements to establish associations between related elements, especially over the Internet. To complement these discovered associations one can attempt to build on relationships based on the respective domain structures and the position and relationships of evaluated elements within those structures. For this purpose, the domains are represented as hierarchical trees in XML syntax; a more general solution handles arbitrary graphs. A structural analysis algorithm builds on associations discovered by other analysis, using these associations to aid in discovering further links that could not have been discovered by purely static examination of the elements and their aggregate content. A number of methodologies are presented by which the algorithm maximizes the number of relevant mappings or associations derived from the XML structures. The paper concludes with comparative results obtained using these three methodologies.

1. Context for Structural Analysis

For effective transfer of information between diverse processes in a networked environment, it is necessary that there be a known and agreed-upon protocol for interconversion between the "languages" they speak. In other words, the data structures they incorporate must agree on how they map between each other. Explicitly requiring the systems to agree on a data format for exchange is not always desirable as it forces constraints on their data structures and induces interdependencies between the systems involved. Enabling interoperability via a static mediator has various drawbacks, as either the mediator or the systems will constantly need to be updated as one of the processes' internal structure changes or new data domains are added. An alternate approach involves more sophisticated mediators that dynamically or statically adapt translations between systems, without interfering with their internal data formats.

A data communication language assumed for the domains is the eXtensible Markup Language (XML) [1]. XML is a flexible meta-language useful as a framework for interoperability: it immediately resolves several issues, such as parsing and character encoding recognition. Documents conforming to valid XML immediately reveal the logical structure of the data they contain, freeing a designer to focus on the analysis itself. XML also incorporates several technologies that can be leveraged in mediating interoperability: these include the XML Linking Language (XLink) [4], which aids in drawing arbitrary associations between resources; XML Schema, a schema description language for XML documents [29, 30, 31]; and XML Stylesheet Language Transforms (XSLT), a language for specifying transformations between XML documents [32]. These tools are very useful in describing and carrying out the process of conversion or translation. What is significantly lacking [2], however, is a process of actually discovering associations between documents and semi-automatically generating the transform scripts which can rearrange and modify the associated data. Requiring constant human intervention at this stage nullifies much of the advantage of having a dynamic, sophisticated mediator.

Our research has concluded that in order to effectively mediate between heterogeneous domains in XML, the following architecture is needed. Given two non-identical domains, two types of applications are required to meet these goals. The first application, the mapper, interrogates the data structures of the source and destination and creates the mappings between them. This application requires some user interaction, since all semantic differences in data cannot be automatically discovered. The second application, the mediator,

operates on instance data to create a tailored product using the mapping rules. The beauty of this design is that once the mapping rules are found, the mediator applications can use the same set of mappings to process the instance data over again, as long as the structures are not changed. And, when the structures do change, the mapper can update the mappings before the mediator begins processing, thus providing current mappings to the mediator for the new instance data. Lastly, since the mediator simply reads in the mapping files, no code changes are necessary to operate on the new data structures. This architecture is depicted in Figure 1.

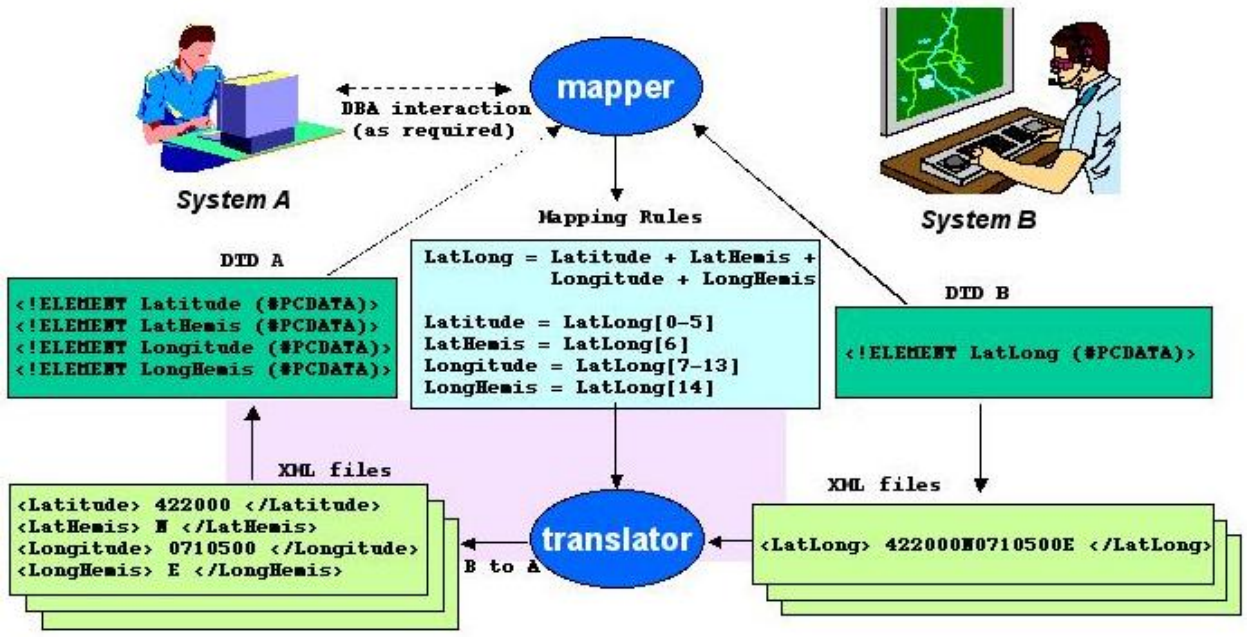

Figure 1. Mapping and Transformation Architecture

In order to research methods for semi-automated semantic correlation between multiple heterogeneous systems, a prototype of the mapper, X-Map, was developed. Key information components required by X-Map are:

1. Knowledge of possible mappings between elements in data structures,
2. The means to represent mappings and store them so they can be reused by other applications, and
3. Transformation functions that perform the specific operations required to transform an XML document from one domain to another.

As part of our research, the following components were developed:

1. An X-Link based association language for representing the mappings,
2. An XSLT based language for representing the transformation functions, and
3. The specification of several different types of mediators that perform various mediation tasks.

This work has been described in [2, 3] and will not be further detailed here.

In X-Map, data about two XML domains are read in, analyzed, and the resulting mappings and transformation functions are generated as output. This design segmented the process of discovering mappings into two separate parts: equivalence analysis, which examines the tags, attributes, and data for similarities in form and content, and structural analysis, which looks at the structures of the XML files for similarities. After analyzing for potential mappings, X-Map displays to the user a graphical representation of the nominations it has made. As part of the User Interface (UI), the user can interrogate the system for information about the nominations, and can then accept or reject any of them. Afterwards, the UI allows the user to add mappings that may have been missed by X-Map. The user also needs to confirm the type of mapping nominated so the proper transformation functions can be created. After all user input is entered, X-Map outputs an X-Map Linkbase (the name given to a linkbase that contains mappings as opposed to hypertext linkings) of the mappings that exist and a set of transformation functions that encode how the mappings are performed. A diagram of the design developed for X-Map is shown in Figure 2.

The technical backbone of X-Map is the automated engine through which associations are discovered and nominated. This is the mechanism via which associations are discovered and captured by the association language. The general case of automatic association is very hard and implies the solving of consistent semantic matching of elements between two data-domains. The goal here is less ambitious: it is the use of semi-automated methods to make consistent semantic matches in suitably constrained cases, and to provide an approach that can make these matches within a specified margin of error with minimal requirement of human intervention. The system will try its best to come up with appropriate semantic matches, and when it cannot provide a match or has serious doubts, or for ultimate validation, it will ask a human operator for assistance. Obviously we would like to minimize the work that has to be performed by the human operator;

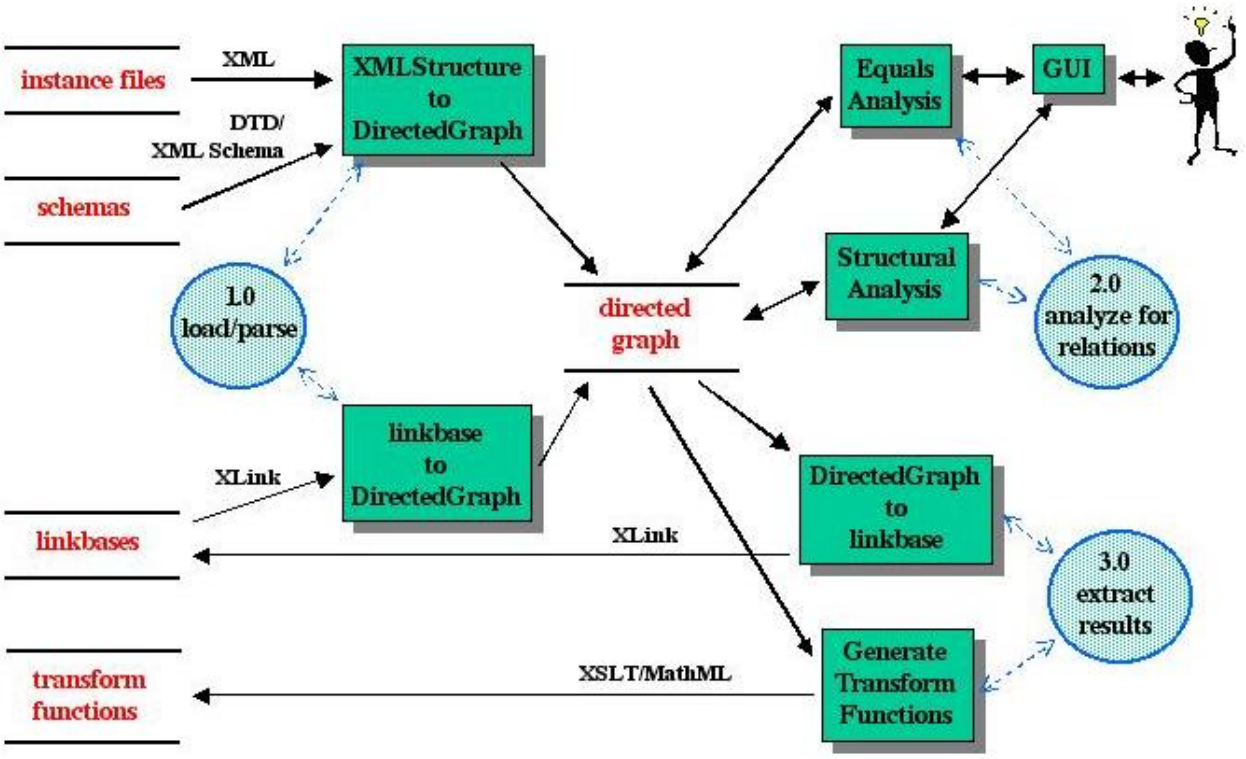

Figure 2. X-Map design

this implies nominating as many genuine mappings as possible while keeping to a minimum the incidence of "false positives" which require an operator to invalidate nominated associations.

Table 1 illustrates some of the common types of associations that may occur, and examples of these. Although some of these use the "equivalence" relationship (defined in [2] as expressing essentially the same physical object or concept, and differing only in representation) as a fundamental building block, others do not. Note that not all of these relationships need be two-way: sometimes a one-way relationship implies an entirely different relationship (or no relationship) in the opposite direction.

In many cases, the nature of the relationship(s) can be read off simply by observing the names and attributes of elements in the disparate domains. The most obvious instance arises when precisely the same element occurs in both domains. Alternately, it may be possible to establish a relationship by scanning for element names in a lookup table: for instance, the X-Map prototype supports using a lexicon of abbreviations, jargon or even inter-language translations. In cases where both of the above approaches fail, the algorithm tries simple string manipulations to compare element names: checking whether one is an acronym for the

Relation Type	XML Example		Description
Equals	`<EmpNo>`	`<EmpNo>`	both the tag names and data are the same
Synonym	`<SSN>`	`<SocialSecNo>`	the tag names are not the same, but the data is
Conversion	`<Weight units="kg">`	`<Weight units="lbs">`	data domain is the same, but a conversion is required
LookUp	`<CountryCode>`	`<CountryName>`	data domain is the same, but some type of code is used
Reordering	`<Date format="ddmmyy">`	`<Date format="mmddyy">`	data domain is the same, but reordering is required
IsLike	`<Title>`	`<JobDescription>`	data from one domain is ”good enough” for the other
Generalization/Specialization			
— SuperType	`<TechnicalPaper>`		data from one domain is a superset of the other
— SubType		`<ConferencePaper>`	data from one domain is a subset of data from the other
Abstraction	`<DailySales>`	`<MonthlySales>`	data from one domain combined into a single value in the other
Aggregation/Decomposition			
— PartOf	`<FName>` `<MName>` `<LName>`		data from one domain is concatenated in the other domain
— Composite		`<Name>`	data from one domain takes multiple fields the other
Defaults	No element exists, always enter “1”	`<Quantity>`	default values needed in the other domain

Table 1. Generic Mappings

other, or simply differently spelled. As the scope of possible nominations increases, one eventually arrives at a point of trade-off between false positives and misses. The notion of a "score" describing a degree of confidence in the nominated association, as illustrated later in this paper, provides an elegant way of handling this trade-off.

The next step up involves examining instances of data in elements being compared. As before, the algorithm uses lookup tables, or scan the data instances for some of the relationships described earlier: precision differences, scaling factors (indicating different units being used), acronyms and abbreviations. There will again exist a trade-off between false positives and misses, depending on the scope of the analysis.

The above two techniques are termed "Equivalence Analysis" and "Data Analysis" respectively. This kind of static analysis is invaluable, especially initially when no other possibility is open. However there will invariably exist relationships which cannot possibly be picked up by the earlier techniques. An instance of this is the "`Employee-SalesPerson`" association described earlier in Table 1. To a human, it is clear that these are related: a `SalesPerson` is an example of an `Employee`. However, there is no semantic relationship between these element names for an automated agent to pick up. It is likely that the data immediately stored under this tag (as opposed to under child tags) are either unrelated, or the relationship does not get picked up by data analysis for whatever reason (e.g. sufficiently different formats). This would be a case where analysis of the data structures comes into play in nominating associations.

It follows that structural analysis and comparison of data domains is a key component in semi-automated detection of semantic mappings. The remainder of this paper describes implementations of this analysis, as tested in the X-Map prototype. The analysis depends heavily on our internal representation of the structure of these domains, so to begin with, this representation is described in greater detail. Subsequent sections describe different modes of resolution of this problem. The fundamental question under study may be approached from several different directions, as follows:

1. As a straightforward problem of developing heuristics for speculating on associations;
2. As a graph theoretic labeling problem: discovering isomorphisms between graphs representing the data structures; or
3. As a search space/optimization problem: minimizing the deviance between speculated and genuine

mappings.

These different ways of looking at the problem shape different approaches toward solving it. Instances of several different types of techniques were implemented to compare and contrast their respective performances on sample data.

2. Internal Representation of Data Structures

The formulation of the data representation is important in order to maximize information that can be derived from it. XML Schemas normally have a highly hierarchical tree-like formation. For the purposes of XML Interoperability, data structure is synonymous with schema since a valid XML document (itself a data receptacle) must conform to some XML Schema. Thus its structure must be known — just parse the schema. Structural information-wise, the two are synonymous. Internal representation of the schema is faithful to its tree structure, additionally storing tag data in child nodes of the nodes corresponding to that tag, and attributes likewise in child nodes of the appropriate tag nodes. Hierarchy edges express hierarchy relationships within data domains; Association edges express relationships between elements across data domains. Formally, the representation $\mathcal{X}$ consists of a tuple of components.

$$\mathcal{X} = \langle E_A, D_1, D_2 ... D_n, s \rangle$$

- Association edges are added in by the engine in the course of its execution; as described they express relationships across domains.

$$E_A \subset \bigcup_{i \neq j} V_i \times V_j$$

- Each domain D_i comprises a set of vertices V_i and hierarchy edges E_{Hi}

$$D_i = \langle V_i, E_{Hi} \rangle = \langle \langle V_{Ti}, V_{Di}, V_{Ai} \rangle, E_{Hi} \rangle$$

  where V_{Ti} represents the set of tags, V_{Di} the tag data, and V_{Ai} the attributes. The vertices are linked by directed edges in E_{Hi}.

$$E_{Hi} \subset (V_{Ti} \times V_{Di}) \cup (V_{Di} \times V_{Ai})$$

- The scoring function $s : E_A \rightarrow \Re$ or $E_A \rightarrow [0, 1]$ is described later.

Both association and hierarchy edges are implicitly asymmetric, $(v_1, v_2) \neq (v_2, v_1)$. For hierarchy edges, this is immediate since direction of the edge is encapsulated in the semantic structure of the schema. For association edges, this captures the earlier observation that not all associations need be two-way and that some are manifestly not so.

Adhering to the tree structure of a valid XML specification imposes a stronger condition. Within each domain D_i there must exist a partial ordering $\leq$ such that $(v, v') \in E_{Hi} \Rightarrow v \leq v'$ (and of course $v, v' \in V_i$). For example, a 3-cycle cannot have such a partial ordering described on it and is, therefore, not a valid domain description.

The above constraint is inherent to XML but may actually be relaxed to our advantage. Notably, the X-Map prototype design does not preclude a "flattening" of the hierarchical tree structure with consolidation of identical nodes. The data structure used (a tagged directed graph) is built to handle any arbitrary graph of tagged nodes. The rationale behind potentially recasting the tree is that for particular structures, a flattened graph will be more computationally palatable. A problem with structural analysis on an open tree is the bias it builds against correlating aggregation and generalization conflicts. In non-trivial cases, these conflicts result in completely different tree structures, which confuse attempts to frame resolutions based on hierarchies. Thus, for our purposes, the schema's "tree" is compacted into a more useful representation.

In either event, this structure manages to preserve the structural information of the XML input, while casting it into a form amenable to analysis. Besides elementary heuristics and more complex AI techniques, casting the data structure into a directed graph as described allows us to leverage the body of existing graph theory work on isomorphism of graphs. Finally, making the distinction between graph edges representing hierarchical associations within domains, as opposed to edges representing mapping-type associations across domains (E_A/E_{Hi}), allows us to represent multiple data domains $D_1, D_2, D_3...$ within a single directed graph. This is important because it enables the association engine to leverage knowledge of mappings between two domains in discovering associations linking them with a third.

Association edges use scores to represent degrees of confidence in nominated associations, rather than a simple on-off distinction. Besides being conceptually simple, this idea is significant in incrementally en-

hancing understanding of the relationship graph. When augmenting information from previous analyses about a speculative relationship, one needs flexibility in the degree to which current knowledge is changed. A regenerative association engine repeatedly iterates through loops of such analyses to further strengthen the scores. As the final step, all that needs to be done is to compare edge scores against a (predetermined) threshold and take appropriate action for edges that cross that threshold. This is in accordance with the notion that one should never discard information, even uncertain information, when trying to establish associations: one never knows when one will need that information again. Confidence level on edges is retained until those edges are proved to not be associations.

Other modes of execution are being explored. Analysis engines can be run "in parallel", and subsequent generated edge scores combined according to a heuristic of choice. The further extension to this concept is to retain separate scores for each of the analysis techniques applied to the schema, and combine and recombine the scores arbitrarily as per valid criteria.

Interpreting the scores as degrees of confidence (probabilities) has the advantage of being conceptually consistent and well-defined. Technically, the more accurate term is "confidence level": the edge definitely is either an association or not, and we are describing the probability of seeing a particular set of sample data *given* that an association exists there. Scores are thereby restricted to the $[0, 1]$ interval, and we draw on probability theory for ideas on how to manipulate them. When applying multiple *atomic* analysis techniques, the weighted average of returned scores is taken for each edge.

The combination of confidence estimates from independent sources was considered in detail. For example, when using the definite-association-vicinity heuristic described in the next section, what needs to be done for a nomination that lies in the vicinity of two or more known associations? Taking any kind of average in this case is inappropriate since it implies that adding a further measure of certainty sometimes *decreases* the level of confidence in the association. A measure is needed that incorporates information from both sources but still retains all properties desired by the scoring mechanism.

Formally, the requirement is for a probability combination function $\alpha : [0, 1]^2 \rightarrow [0, 1]$ that satisfies the following conditions:

- Probabilities must lie between 0 and 1, obviously.

$0 \leq \alpha(x, y) \leq 1$ for $0 \leq x, y \leq 1$

- Order should not be relevant, so α should be both associative and commutative.
  $\alpha(x, y) = \alpha(y, x)$ and $\alpha(x, \alpha(y, z)) = \alpha(\alpha(x, y), z)$
- Adding confidence levels never decreases the level of confidence.
  $\alpha(x, y) \geq \max(x, y)$
- Adding confidence 0 has no effect.
  $\alpha(x, 0) = x$
- Adding confidence 1 indicates complete confidence in the mapping, unaffected by other estimations.
  $\alpha(x, 1) = 1$

A little inspection shows that any function of the form

$$\alpha(x, y) = \beta^{-1}(\beta(x) + \beta(y))$$

will suffice, where β is a strictly increasing function such that $\beta(0) = 0, \beta(1) = \infty$. For example, using an exponential function like a sigmoid, $\beta(x) = e^x/(1 - e^x)$ gives

$$\alpha(x, y) = (x + y - 2xy)/(1 - xy)$$

However, using the probability notion and regarding the two influencing associations as 'events' each of which can independently affect the edge in question yields the even simpler formula

$$\begin{aligned} \beta(x) &= \log(1 - x) \\ \alpha(x, y) &= 1 - (1 - x)(1 - y) \\ &= x + y - xy \end{aligned}$$

This simpler function was selected for purposes of the current analysis.

3. Heuristics Based Approach

Discovering mappings between elements in an XML environment can be tricky because the hierarchy is not guaranteed to contain any "meaning" for associated elements. Fortunately, in this environment, the hierarchy does usually embed information about its constituent elements. This is domain knowledge specific to our purpose that will be exploited. In the case of heuristics, the engine uses patterns and functions to derive information from structural associations. Many of these actually depend on prior known mappings — they need to be "bootstrapped" by first running Equivalence Analysis and Data Analysis on the domains in question, or reading off a set of known associations from an XMap Linkbase [2].

The simplest case involves direct hierarchical associations. Thus if $A \rightarrow B \rightarrow C$ and $D \rightarrow E \rightarrow F$ are pieces of the hierarchy in two distinct domains, and a strong mapping exists between B and E, one can speculate with some (perhaps low) degree of confidence that C and F are related. One also acquires (slightly higher, but still low) confidence that A and D are related, or their parents.

Next, note that the converse is somewhat stronger: If *both* $A - D$ and $C - F$ are related, there is a relatively higher degree of confidence that B and E are related. In some sense, the nearby associations at the same points in the hierarchy reinforce confidence in the association currently being examined. If the known associations are fairly distant, however, the level of confidence in these two elements being related drops off rapidly. Based on experiments with sample data, this drop-off was found to be roughly exponential, with best results coming from a discount factor of around 0.65 for the dataset that was analyzed. Of course, the set of actual bests will depend on the precise nature of the data domains being examined. A good avenue for further exploration would be to dynamically increase or decrease this factor based on the characteristics of the domain graphs.

The score-combining function from the previous section turns out to be useful in implementing this heuristic. Under this scheme, potential edges are rated based on the proximity of their end nodes to the end nodes of known associations. The greater the incidence that such known associations lie within the vicinity of the speculated edge, the more the edge score gets reinforced. Nodes are scored based on both proximity and direction with respect to known associations. Thus for instance in the example above, if $B - E$ is a mapping, then $A - D$ should be rated more highly than $A - F$. The latter is still possible however. There might have

been ambiguity at some stage in which node is the parent and which one the child. Similarly, A's other child – D's other child will be rated more highly, and so on. Associations of the form $A - F$ are not ignored, but acquire lower scores; this flexibility is provided in evaluating how their relationship is affected.

Other methods used include deriving associations in cycles (but this necessitates having run the graph through the *flattening* step described in the previous section: obviously a tree has no cycles). Directed cycles often indicate the presence of aggregation and generalization conflict due to similar "sorts" of information organized or aggregated differently for different data-domains. The resolution of these cycles can frequently indicate associations that would not otherwise have been detected. Finally, associations can usefully be "chained" . The concept is highly intuitive: given three elements A, B, C in distinct domains, a mapping relationship between A and B, as well as one between B and C, is strongly indicative of a potential relationship between A and C. If intermediate associations are not definite but speculative, the idea becomes cloudier, but one can still use probability-combination type formulae of the type described in the preceding section. What makes this possible is the utilized mode for storing multiple data domains in a single graph representation.

4. Graph Theory Based Approach

In order to identify and analyze semantic interdependencies in a complex XML environment, one powerful approach is to cast the structural analysis problem in terms of graph theory. The underlying problem of finding equivalencies in (unlabeled) graphs has been deeply analyzed computationally [7], and we build upon that research with reference to our particular operational environment. For certain classes of graphs, solving graph isomorphism is provably NP-complete [13, 9]. Subgraph isomorphism, i.e. finding the largest isomorphic subgraphs of given graphs, is also a famous NP-complete problem [8]. There are known randomized algorithms [6, 5] for graph isomorphism that run faster; while these take less time, typically $O(n^2)$, they only work on a subset of the cases. The algorithms still need to be adapted to apply to our problem, for the following reasons:

1. One is not trying to determine perfect isomorphism, just "likely" links.

2. During analysis, some of the nodes will already be labeled in the sense that some of their mutual

correspondences are already known.

3. Our problem is much harder, because the graphs in XML problems are not perfectly isomorphic, but merely "similar". There will always be some nodes with no correspondences in the other domain, as well as nodes for which the hierarchy and the parenthood structure are not matched by their analogs on the other side.

We describe a domain (i.e. a graph G) as a set V of vertices and a set E of edges. Given two domains $G_1 = \langle V_1, E_1 \rangle$ and $G_2 = \langle V_2, E_2 \rangle$, the problem is to find $V_1' \subset V_1$ and $V_2' \subset V_2$ such that the graphs induced by V_1' and V_2' on G_1 and G_2 are isomorphic. Also $|V_1'|$ or $|V_2'|$ is to be as large as possible. The bijection $f : V_1' \rightarrow V_2'$ is simply to be expressed in the association edges: our X-Map structure $\mathcal{X}$ contains E_A such that $f(v_1) = v_2 \Leftrightarrow (v_1, v_2) \in E_A$. For isomorphism, one must also have $(v_1, v_2) \in E_1 \Leftrightarrow (f(v_1), f(v_2)) \in E_2$.

The maximum *bounded* common induced subgraph problem (MAX-CIS) is the same problem with restricted space of input instances; so now G_1, G_2 are constrained to have degree at most B for some constant B [7]. Bounding degrees by a constant factor is a realistic assumption for human-usable XML files, so it is worthwhile to also consider the (perhaps easier) problem of MAX-CIS. Unfortunately, MAX-CIS is also provably NP-complete; [7] gives a reduction to the well-known MAX-CLIQUE problem.

Approaches proposed by other researchers to attack this problem include NAUTY [10], the Graph Matching Toolkit [11], and Combinatorica [12]. Based on our target application, a different approach was implemented as described in the following paragraphs.

A logical way to address the problem, aided by definite knowledge of some associations, is to guess the correspondence of some k vertices in one domain with k in the other, and see what the assumption of these correspondences implies. (Of course corresponding vertices would need to have the same or "nearly the same" degree). Depending on the initial assignment of k nodes, if one looks at powers of the adjacency matrix whose rows are the nodes that have been fixed, these matrices provide a classification of the columns that will help decompose them adequately to distinguish them, or make groups of possibly similar nodes much smaller than before. The higher the power of the adjacency matrix, the more "distanced" the classification is from the original nodes and in the non-identical approximation case, the less reliable it will be.

Evaluating the efficiency of the above procedure is not easy. In the worst case, one needs to deal with graphs that look like projective planes, for which it would be very unlikely to choose the correct k nodes in a reasonable number of tries. In the practical case one could use a probabilistic argument averaging over the class of graph problems derived from XML trees. While there would be some subjectivity in describing our general data structures in this numerical fashion, this would not be excessive as XML graphs are characterized by a restricted structure.

The X-Map prototype currently incorporates an implementation of the algorithm described above.

Although not used in the prototype, one alternative strategy was analyzed in detail. Erdös and Gallai's classic extremal function [14] gives the size of the smallest maximum matching, over all graphs with n vertices and m edges. This is the exact lower bound on $\gamma(G)$, the size of the smallest matching that a $O(m + n)$ time greedy matching procedure may find for a given graph G with n vertices and m edges. Thus the greedy procedure is asymptotically optimal: when only n and m are specified, no algorithm can be guaranteed to find a larger matching than the greedy procedure. The greedy procedure is in fact complementary to the augmenting path algorithms described in [17]. The greedy procedure finds a large matching for dense graphs, while augmenting path algorithms are fast for sparse graphs. Well known hybrid algorithms consisting of the greedy procedure followed by an augmenting path algorithm execute faster than the augmenting path algorithm alone [15].

An initial matching for a greedy maximum cardinality matching algorithm can be constructed using the following procedure [17]:

- Start with the empty matching;
- Repeat the following step until the graph has no edges:
  - remove all isolated vertices
  - select a vertex v of minimum degree
  - select a neighbor w of v that has minimum degree among v's neighbors
  - add $\{v, w\}$ to the current matching
  - remove v and w from the graph.

In the worst case, the greedy procedure performs poorly [15]. For $r \geq 3$, a graph D_r of order $4r + 6$ can be constructed such that the greedy procedure finds a matching for D_r only about half the size of a maximum matching [18]. This performance is as poor as any procedure that attempts to find a maximal matching.

There are classes of graphs for which the greedy procedure always finds a maximum matching [18]. Using a straightforward kind of priority queue that contains one bucket for each of the n possible vertex degrees, the greedy procedure can be made to run in $O(m + n)$ time and storage for a given graph with n vertices and m edges [16]. The $O(m + n)$ running time is asymptotically faster than the fastest known maximum matching algorithm for general graphs or bipartite graphs [19, 20, 21, 22, 23, 24, 25, 26]. Areas that warrant enhancement include: the need to support broader subsets of graphs; operation using $O(m + n)$ time and storage; reduction in overhead; and reduction of computational complexity.

The matching found by the greedy procedure will depend on how ties are broken. Let $\gamma(G)$ be the size of the smallest matching that can be found for a given graph G by the greedy procedure, i.e., $\gamma(G)$ is the worst case matching size, taken over all possible ways of breaking ties.

It is proved in [15] that any graph G with n vertices and $m \geq 1$ edges satisfies

$$\gamma(G) \geq \min(\left\lfloor n + \frac{1}{2} - \sqrt{n^2 - n - 2m + \frac{9}{4}} \right\rfloor, \left\lfloor \frac{3}{4} + \sqrt{\frac{m}{2} - \frac{7}{16}} \right\rfloor). \quad (1)$$

This bound is the best possible — when only n and m are given, no algorithm finds matchings larger than that found by the greedy procedure. The analysis in [16] proves the simpler but looser bound of $\gamma(G) \geq m/n$.

The bound above may be used alone, or in conjunction with augmenting path algorithms — the fastest known algorithms for finding a maximum matching. All known worst-case time bounds for augmenting path algorithms are $\omega(m + n)$. Traditionally, one uses a hybrid algorithm: first, use the greedy procedure to find a matching M in $O(m + n)$ time; then, run an augmenting path algorithm with M as the initial matching. Intuitively, if the input graph is dense, the greedy procedure finds a large matching, and the augmenting path algorithm needs only a few augmentation phases. Conversely, if the input graph is sparse, then each augmentation phase is fast. As such, one can use one kind of method (such as the greedy procedure) for handling dense graphs, and another kind of method (like an augmenting path algorithm) for handling other graphs.

We consider the problem of finding a maximum matching, with $G = \langle V, E \rangle$ being the graph representing a domain as before, with $E = E_H$. Use vw as an abbreviation for an edge $\{v, w\} \in E$. Then $\forall v \in V$, the graph $G - v$ is the graph with vertex set $V - v$, and edge set $\{xy \in E : x \neq v, y \neq v\}$. Let the number of vertices and edges in G are respectively $n(G)$ and $m(G)$, and the degree of a minimum degree vertex of G be denoted $\delta(G)$. The matching number of G is denoted by $\nu(G)$, i.e., $\nu(G)$ is the size of a maximum matching for G. The complete graph on n vertices is K_n.

Erdös and Gallai [14] demonstrate an exact lower bound on the discrepancy between the maximum matching discovered by the greedy algorithm and a genuine matching which may exist. Accordingly, the greedy algorithm could be a good starting point for discovering mappings. It can also be augmented with a hybrid approach, or the user could plug in the adjacency matrix techniques discussed earlier.

Incidentally, the maximum bounded common induced subgraph problem can be trivially approximated to within a $(B + 1)$ factor [7]. Just pick independent sets V_1', V_2' by adding nodes (in any order) which are unconnected to any node currently in the set. This will yield a pair of independent sets whose collective size is at least $\min(|V_1|, |V_2|)/(B + 1)$: this is because for each node either it itself or one of its neighbors must have been picked to add to the sets. Obviously the optimal solution has size $\leq \min(|V_1|, |V_2|)$, so we have approximated to within a constant factor. This approach is good for small graphs and may even yield local maxima; however it fails in comparison to the other starting algorithms referred to above.

There is a close connection between the problems of subgraph isomorphism described above, and the problem of semantic entity correlation in XML schema. The graphical structure of the schema can facilitate entity correlation, and complement other mechanisms to attain the latter goal. The reason is that closely related elements across domains are tied to closely related structures in the graphical hierarchy. Accordingly, an algorithm which efficiently discovers graph isomorphisms will also efficiently suggest XML entity correspondences.

5. Artificial Intelligence Based Techniques

If one looks at the problem as a search space question, there are a number of relevant AI techniques which can be brought into play to handle it. The use of continuous scores opens up a vast number of mathematical

operations that can be performed on sets of edges. When one brings into play feedback loops that repeatedly update the scores of edges over several cycles, one can apply neural nets, Bayesian analysis and other technologies which will also take into account existing data and requisite domain knowledge.

Neural Networks (NNs) are one of the avenues chosen for examination. NNs learn from past performance to determine candidates for the current session [28]. In the case of NNs, a description of the subgraph around a potential edge serves as input to the NN. An interesting feature/weakness of techniques of this type is the degree to which they use the "past to predict the future". They need to be trained on test data before being used on fresh domains; this means that if the training data are significantly different from the actual data in target domains, the results posses little relevance. The use of these techniques thus makes some assumptions about the validity of past training data with respect to current resources.

Arguably, the greatest value of neural networks lies in their pattern recognition capabilities. Neural networks have advantages over other artificial intelligence techniques in that they allow continuous, partial and analog representations. This makes them a good choice in recognizing mappings, wherein one needs to exploit complex configurations of features and values.

Clifton and Li [27] describe neural networks as a bridge across the gap between individual examples and general relationships. Available information from an input database may be used as input data for a self-organizing map algorithm to categorize attributes. This is an unsupervised learning algorithm, but users can specify granularity of categorization by setting the radius of clusters (threshold values). Subsequently, back-propagation is used as the learning algorithm; this is a supervised algorithm, and requires target results to be provided by the user. Unfortunately, constructing target data for training networks by hand is a tedious and time-consuming process, and prone to pitfalls: the network may be biased toward preferring certain kinds of mappings, depending on how the training data is arranged.

Characteristics of schema information, such as attributes, turn out experimentally to be very effective discriminators when using neural nets to determine tag equivalence. The policy of building on XML turns out to be very useful: schema information is always available for a valid document. Furthermore, our scoring mechanism for speculated edges lends itself well to the continuous analog input/output nature of neural networks.

6. Results

In the preceding sections, we examined three broad categories of approaches to perform structural analysis on XML formatted data between similar but non-identical domains — heuristic based approaches; graph theoretic approaches; and AI based approaches. Having implemented instances of each of these, it was instructive to compare the performance on our prototype application, to see which ones yield promising results. The nature of the problem requires that human validation be involved in evaluating success: specifically, the technique must be executed on data for which a human has already specified which are and which are not viable mappings. An algorithm can then be appraised based on a metric that takes into account both correctly located connections (positive) and false connections incorrectly marked (negative). Comparative results are shown in Table 2.

These scores were taken on test domains consisting of sample XML files based on real-world data. One issue which arises in evaluating techniques is that structural analysis is a fairly new approach to entity correlation. There is little comparative information from other researchers regarding alternative approaches.

	# actual mappings nominated	# non-mapping pairs nominated	Score (%correct - %incorrect)
Total	13	245	
Bootstrapped (already known)	4	0	
Number to be detected	9	0	
Heuristics	7	10	73
Adjacency matrices	5	15	49
Neural networks	5	37	39

Table 2. Performance of Structural Analysis Techniques

Overall heuristic approaches performed significantly well, both in determining valid matches and in avoiding fake ones. Not only are these methods straightforward to conceive and implement, they also offer high transparency. It is easy to understand how the internal settings affect final results; this facilitates the modification of parameters to maximize effectiveness of the algorithm for a particular purpose. In contrast, more sophisticated alternatives tended to be “black boxy” in that it is difficult to relate the internals of the process

to its final outcome. Heuristic methods showed errors of under 30% on our restricted data sets.

The adjacency matrix implementation was partially transparent, but did not yield as low an error rate. The worst cases arose on sub-schema with a relatively linear structure, or when adjacent nodes in one hierarchy were displaced by having some redundant node between them in another hierarchy. Incidentally, the graph theoretic analysis turned out to require less "bootstrapping" by other analyses than either heuristics or neural nets. The analysis draws more information from the schema structure than from prior known associations across the domains.

Neural networks turned out to be unpredictable and difficult to modulate for our restricted datasets. However, this should not be taken to mean that this option is impractical for the structural analysis problem as a whole. The real power of neural networks and other more elaborate techniques shows up on inputs of greater size and complexity than was reflected in our tests. Since they acquire knowledge cumulatively (rather than recomputing everything on a per-analysis basis) they also show better results on a longer timescale. Due to this characteristic, a neural networks based approach has potential for further investigation.

7. Conclusion

In semi-automatically discovering semantic links between elements of heterogeneous domains, such as a cross-internet application, structural analysis of the domains can serve as an invaluable aid. This paper has outlined several of the possible approaches for performing such analysis. The proposed approaches have been incorporated into the XMap prototype system of an automated association engine. Our current efforts are geared toward performing quantitative comparisons between these analytic techniques, in order to determine which might yield the best results in mediating between non-identical data domains. The objective of such an analysis is to develop an integrated approach that can automatically adopt the optimal approach tailored for the particular set of data, the particular source of the data, and the desired characteristics for the user and the data.

8. Acknowledgments

We thank a number of colleagues at MITRE and MIT who were involved in the development of various aspects of this endeavor.

References

[1] C. S.-M. T. Bray, J. Paoli and E. Maler. http://www.w3.org/TR/REC-xml, October 2000. eXtensible Markup Language (XML) 1.0, W3C recommendation.

[2] D. Wang. Automated semantic correlation between multiple schema for information exchange. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, 2000.

[3] E. Byon. X-Map Linkbase Search Engine (XLSE): A Search Engine for XML Documents. Draft Manuscript, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, 2002.

[4] XML Linking Language (XLink) Version 1.0 http://www.w3.org/TR/xlink/, 27 June 2001

[5] M. M. Halldorsson and K. Tanaka. Approximation and special cases of common subtrees and editing distance. In *Proc. 7th Ann. Int. Symp. on Algorithms and Computation*, pages 75–84. Springer, 1996. Lecture Notes in Computer Science, number 1178.

[6] V. Kann. *On the approximability of NP-complete optimization problems*. PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, 1992.

[7] V. Kann. On the approximability of the maximum common subgraph problem. In *Proc. 9th Annual Symposium on Theoretical Aspects of Computer Science*, pages 377–388. Springer, 1992. Lecture Notes in Computer Science, number 577.

[8] M. R. Garey, D. S. Johnson. Computers and Intractability: a guide to the theory of NP-completeness. W.H. Freeman and Company, San Francisco, 1979.

[9] C. Lin. Hardness of approximating graph transformation problem. In *Proc. 5th Ann. Int. Symp. on Algorithms and Computation*, pages 74–82. Springer, 1994. Lecture Notes in Computer Science, number 834.

[10] B. D. McKay. Nauty. http://cs.anu.edu.au/people/bdm/nauty/, February 2000.

[11] GMT - Graph Matching Toolkit http://www.cs.sunysb.edu/ algorith/implement/gmt/ implement.shtml, July 1999

[12] S. S. Skiena. *Implementing Discrete Mathematics: Combinatorics and Graph Theory in Mathematica*. Addison-Wesley, Redwood City CA, June 1990.

[13] K. Zhang and T. Jiang. Some MAX SNP-hard results concerning unordered labeled trees. *Inform. Process. Lett. 49, 249-254*, June 1994.

[14] Paul Erdös and T. Gallai. On maximal paths and circuits of graphs. *Acta Math. Acad. Sc. Hungar.*, 10:337–356, 1959.

[15] Andrew Shapira An Exact Performance Bound for an O(m+n) Time Greedy Matching Procedure Electronic Journal of Combinatorics Paper R25 of Volume 4(1)

[16] Andrew Shapira. Matchings, degrees, and O$(m + n)$ time procedures, in preparation.

[17] B. M. E. Moret and H. D. Shapiro. *Algorithms from P to NP*. The Benjamin/Cummings Publishing Company, Inc., 1991.

[18] Andrew Shapira. Classes of graphs for which an O$(m + n)$ time greedy matching procedure does and does not find a maximum matching, in preparation.

[19] H. Alt, N. Blum, K. Mehlhorn, and M. Paul. Computing a maximum cardinality matching in a bipartite graph in time O$(n^{1.5}\sqrt{m/\log n})$. *Information Processing Letters*, 37:237–240, February 1991.

[20] Norbert Blum. A new approach to maximum matching in general graphs. In *ICALP 90 Automata, Languages and Programming*, pages 586–597, Berlin, July 1990. Springer.

[21] Tomás Feder and Rajeev Motwani. Clique partitions, graph compression, and speeding-up algorithms. In Baruch Awerbuch, editor, *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing*, pages 123–133, New Orleans, LA, May 1991. ACM Press.

[22] Zvi Galil. Efficient algorithms for finding maximum matchings in graphs. *Computing Surveys*, 18:23–38, 1986.

[23] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4), December 1973.

[24] L. Lovász and M. D. Plummer. *Matching Theory*, volume 121 of *North-Holland mathematics studies*. North-Holland, Amsterdam, 1986.

[25] S. Micali and V. V. Vazirani. An O$(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximal matching in general graphs. In *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science*, pages 17–27. IEEE Computer Society Press, 1980.

[26] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice Hall, 1982.

[27] Wen-Syan Li and Chri Clifton. Semantic Integration in Heterogenous Databases using Neural Networks Proceedings of the 20th VLDB Conference, 1994

[28] Patrick Henry Winston Artificial Intelligence Addison-Wesley Co., 1992

[29] C. M. Sperberg-McQueen and Henry Thompson. XML Schema. W3C Architecture Domain, April 2000. http://www.w3.org/XML/Schema

[30] H.S. Thompson, D. Beech, M. Maloney, N. Mendelsohn, editors. XML Schema Part 1: Structures. W3C Recommendation, May 2, 2001. http://www.w3.org/TR/xmlschema-1

[31] P.V. Biron, A. Malhotra, editors. XML Schema Part 2: Datatypes. W3C Recommendation, May 2, 2001. http://www.w3.org/TR/xmlschema-2

[32] J. Clark, editor. eXtensible Stylesheet Language Transform (XSLT) Version 1.0. W3C Recommendation, November 16, 1999. http://www.w3.org/TR/xslt