

# Discovering Entity Correlations between Data Schema via Structural Analysis

by

Ashish Mishra

Submitted to the Department of Electrical Engineering and Computer Science  
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Computer Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2002

© Massachusetts Institute of Technology 2002. All rights reserved.

Author .....  
Department of Electrical Engineering and Computer Science  
May 24, 2002

Certified By .....  
Dr. Amar Gupta  
Co-Director, Productivity From Information Technology (PROFIT) Initiative  
Thesis Supervisor

Accepted by .....  
Arthur C. Smith  
Chairman, Department Committee on Graduate Students



# **Discovering Entity Correlations between Data Schema via Structural Analysis**

by

Ashish Mishra

Submitted to the Department of Electrical Engineering and Computer Science  
on May 24, 2002, in partial fulfillment of the  
requirements for the degrees of  
Bachelor of Science in Computer Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

## **Abstract**

At the forefront of data interoperability is the issue of semantic translation; that is, interpretation of the elements, attributes, and values contained in data. Systems which do not adhere to pre-defined semantics in their data representations need to dynamically mediate communication between each other, and an essential part of this mediation is structural analysis of data representations in the respective data domains. When mediating XML data between domains, one cannot rely solely on semantic similarities of tags and/or the data content of elements to establish associations between related elements. To complement these associations one can build on relationships based on the respective domain structures, and the position and relationships of evaluated elements within these structures. A structural analysis algorithm uses associations discovered by other analysis, discovering further links which could not have been found 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 XML structures. The paper concludes with comparative results obtained using these methodologies.

Thesis Supervisor: Dr. Amar Gupta

Title: Co-Director, Productivity From Information Technology (PROFIT) Initiative



# Acknowledgments

My supervisor at MITRE, Michael Ripley, was a huge contributor throughout this project — he was a voice for suggestions and criticism, an ear to bounce ideas off, and a push at the right times when the going got slow. Additionally, my thesis supervisor, Dr. Amar Gupta, gave both direction and support in all phases of my research. Much of this thesis draws from a paper the three of us co-authored in 2002.

The foundation for the X-Map project was laid by David Wang, whose thesis was also a source for much of my work. My colleague, Eddie Byon, was a major contributor to the coding process, as well as helpful in discussing ideas and more mundane matters such as transportation.

Other helpers and motivators included Doug Norman at MITRE; Brian Purville, Danny Lai, and Andrew Chen at MIT; and Anne Hunter, the wonderful department administrator. Most of all, I owe this to my parents, without whom none of this would have been remotely possible.



# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Context for Structural Analysis</b>            | <b>11</b> |
| 1.1      | X-Map . . . . .                                   | 12        |
| 1.2      | The Association Engine . . . . .                  | 15        |
| <b>2</b> | <b>Internal Representation of Data Structures</b> | <b>21</b> |
| 2.1      | XML Schema . . . . .                              | 21        |
| 2.2      | Scoring Mechanism . . . . .                       | 24        |
| <b>3</b> | <b>Means of Analysis</b>                          | <b>27</b> |
| 3.1      | X-Map Design and Rationale . . . . .              | 28        |
| 3.1.1    | Enumeration of Relation Classes . . . . .         | 31        |
| 3.1.2    | Discovering Relations . . . . .                   | 35        |

|          |  |           |
|----------|--|-----------|
| 3.1.3    | X-Map's Strategies . . . . .                         | 37        |
| 3.1.4    | Regenerative Association Engine . . . . .            | 41        |
| 3.2      | Heuristics . . . . .                                 | 42        |
| 3.3      | Graph Theory Analysis . . . . .                      | 43        |
| 3.4      | Other Techniques . . . . .                           | 45        |
| <b>4</b> | <b>Heuristics Based Approach</b>                     | <b>47</b> |
| 4.1      | Reasoning from Associations . . . . .                | 49        |
| 4.2      | Associations in Cycles . . . . .                     | 50        |
| 4.3      | Chaining Associations across Domains . . . . .       | 50        |
| <b>5</b> | <b>Discovering Mappings using Graph Theory</b>       | <b>53</b> |
| 5.1      | Relation to Graph Problems . . . . .                 | 53        |
| 5.2      | Solving Isomorphism via Adjacency Matrices . . . . . | 54        |
| 5.3      | Other Avenues . . . . .                              | 56        |
| 5.4      | Definitions and Notation . . . . .                   | 59        |
| 5.5      | A Related Theorem . . . . .                          | 59        |
| 5.6      | The Hybrid Approach . . . . .                        | 61        |

|          |   |           |
|----------|---|-----------|
| <b>6</b> | <b>Artificial Intelligence Based Analysis — Neural Nets</b> | <b>65</b> |
| 6.1      | Neural Nets . . . . .                                       | 65        |
| 6.2      | NNs in X-Map . . . . .                                      | 67        |
| 6.3      | Implementation . . . . .                                    | 68        |
| <b>7</b> | <b>Results</b>  | <b>71</b> |
| 7.1      | Technique Performances . . . . .                            | 71        |
| 7.2      | Conclusion . . . . .  | 74        |
| <b>A</b> | <b>Source Code</b>  | <b>75</b> |



# Chapter 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) [33]. 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) [42], which aids in drawing arbitrary associations between resources; XML Schema, a schema description language for XML documents [58, 59, 60]; and XML Stylesheet Language Transforms (XSLT), a language for specifying transformations between XML documents [61]. These tools are very useful in describing and carrying out the process of conversion or translation. What is significantly lacking [34], 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.

## **1.1 X-Map**

Our research [5] 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 1-1 [5].

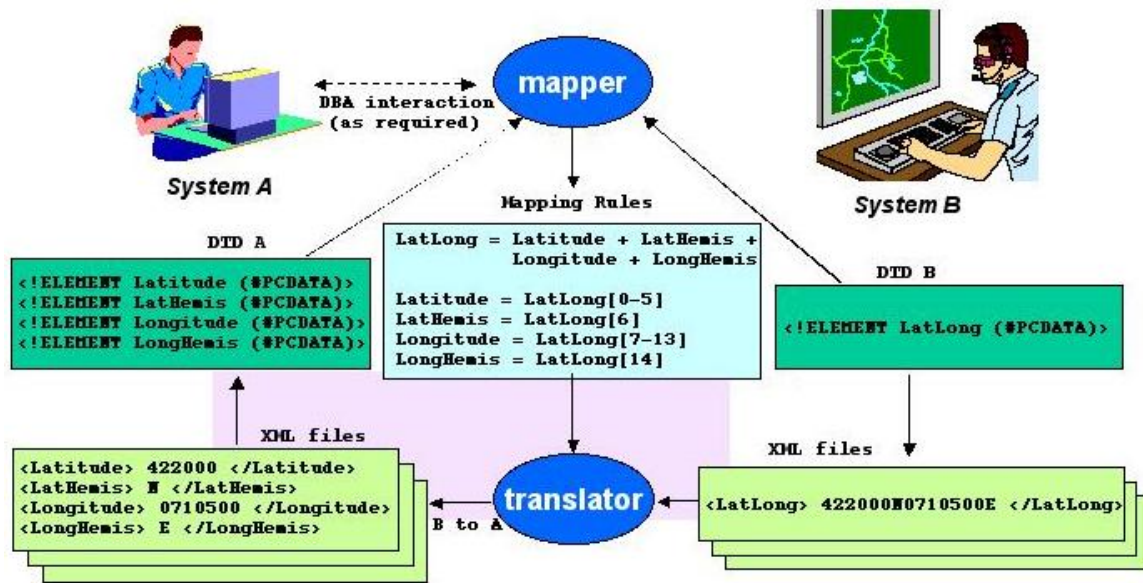


Figure 1-1: Mapping and Transformation Architecture

The X-Map system for semi-automated semantic correlation seeks to mediate XML data interoperability between multiple heterogeneous systems. X-Map would play a significant role in the Joint Battle InfoSphere (JBI), which aims to provide information superiority to US Armed Forces. The JBI adheres to a publish and subscribe mechanism, which facilitates the sharing of information between interested parties, manipulation of information to create knowledge, and dissemination of that new knowledge to other interested parties upon creation. The interested parties send information via messages asynchronously back and forth. The role that fuselets play in the JBI amounts to enhancing and automating all three functionalities of the JBI; Information Input, Information Manipulation, and Knowledge Output and Interaction. That is, fuselets fuse information from several subscribed sources into knowledge.

X-Map's essential design was developed as part of the 'Fuselet Architecture for Adaptive Information Structures' project at MITRE corporation. The goal of the X-Map project is to enable adaptive data interoperability between participating systems through a transparent mediator. Among the key information components required for this task are (a) knowledge of possible mappings between elements in data structures, (b) a semi-automated means to derive these mappings, as well as to represent and store them so they can be reused by other applications, (c) transform functions to facilitate the mediation of data between systems, given knowledge of mappings between them, as well as to represent the transformations done by the mediation so they can be easily reused. X-Map's role is in the the first two tasks; semi-automatically discovering mappings and relationships, as well as storing information about them.

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 [5], the following components were developed:

1. An XLink 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 [34, 35] 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 1-2 [5].

## **1.2 The Association Engine**

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 re-

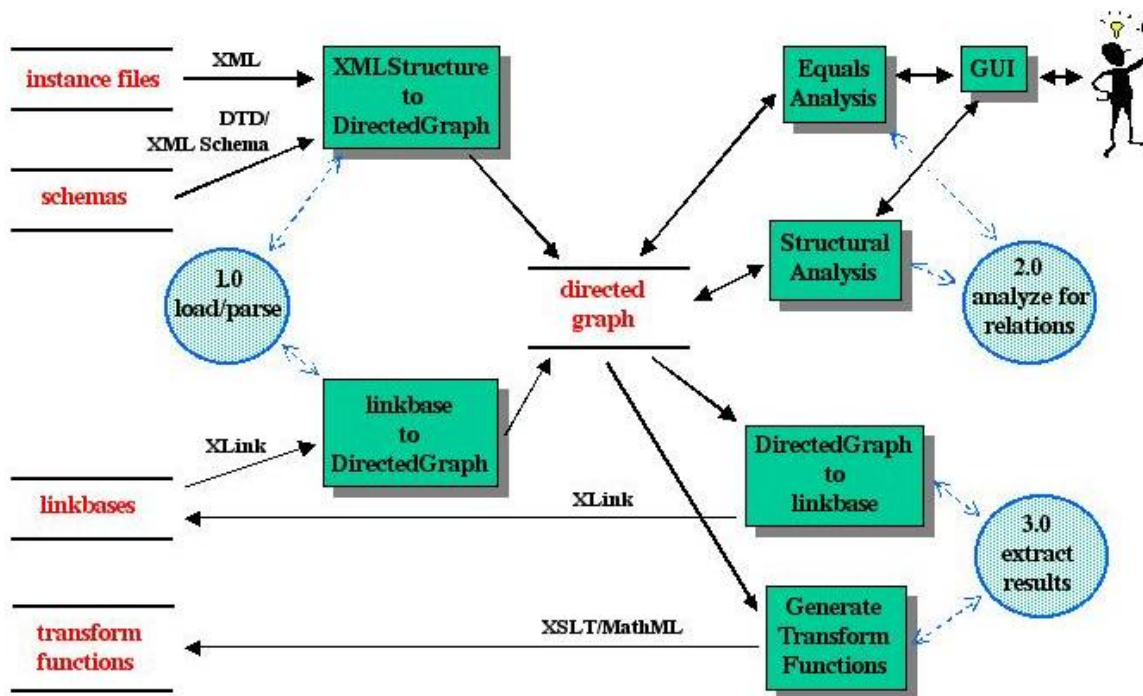


Figure 1-2: X-Map design

quirement 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; 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 1.1 [5] 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 [34] 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.

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

Table 1.1: Generic Mappings

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 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 scans 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 1.1. 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.



## Chapter 2

# Internal Representation of Data Structures

### 2.1 XML Schema

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.

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

Where the components are defined as follows:

- $E_A$  is the set of association edges, added by the engine in the course of its execution; as described they express relationships across domains.

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

- $D_1, D_2, \dots$  are domains, comprising a set of vertices  $V_i$  and hierarchy edges  $E_{H_i}$

$$D_i = \langle V_i, E_{H_i} \rangle = \langle \langle V_{T_i}, V_{D_i}, V_{A_i} \rangle, E_{H_i} \rangle$$

where  $V_{T_i}$  represents the set of tags,  $V_{D_i}$  the tag data, and  $V_{A_i}$  the attributes. The vertices are linked by directed edges in  $E_{H_i}$ .

$$E_{H_i} \subset (V_{T_i} \times V_{D_i}) \cup (V_{D_i} \times V_{A_i})$$

- The scoring function  $s : E_A \rightarrow \mathfrak{R}$  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  $(v, v') \in E_{H_i} \Rightarrow v \leq v'$  (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 ( $E_A/E_{H_i}$ ), allows us to represent multiple data domains  $D_1, D_2, D_3\dots$  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 enhancing 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.

## 2.2 Scoring Mechanism

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 closed  $[0, 1]$  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.

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

- Order should not be relevant, so  $\alpha$  should be both associative and commutative.

$$\alpha(x, y) = \alpha(y, x) \text{ and } \alpha(x, \alpha(y, z)) = \alpha(\alpha(x, y), z)$$

- 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  $\beta$  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.

# Chapter 3

## Means of Analysis

Given the framework described in the preceding pages, the main thrust of my thesis involves implementing and evaluating different sets of techniques for studying the graph structures. In investigating the feasibility and efficiency of this technique, it would be necessary to work with a significant amount of real-world data to determine the parameters of the directed graphs involved, and figure out the best approach. The techniques are described below.

Basically, X-Map's job is to discover new semantic relations semi-automatically given schemas that correspond to data in the relevant data-domains. Then, it should deposit the knowledge in a reusable format (i.e. in the Association Language) that captures the semantic notions of association and the appropriate transformation function description (i.e. the Transformation Language), if any, for other applications such as the Mediator or fuselets to use. Finally, an illustrative prototype such as the Mediator will show how to use the knowledge in the associations to help it mediate (such as translating and transforming) data between data-domains to achieve XML Interoperability.

It is desirable to uncouple X-Map and the Mediator because the tasks they perform are fundamentally different — the former creates associations while the latter consumes them. That means that it is also a good design choice to abstract the Association and Transformation Languages away from both X-Map and the Mediator since then they share the languages as an interface, which minimizes dependencies between the two modules. Furthermore, the uncoupling allows the Mediator to invoke X-Map on-demand to process XML schemas as it runs across new ones, and it also allows X-Map to run silently in the background and use idle processor moments to add to its known relations linkbase via the regenerative association engine.

### **3.1 X-Map Design and Rationale**

Operationally speaking, X-Map performs the following tasks, in order: [34]

1. Locate and read in the desired schemas to associate (either specified by the XML documents or given as-is) via the prescribed programmatic XML interface<sup>1</sup>.
2. Using a heuristic, determine which analysis to use (including all) to generate the associations.
3. Generate the associations and improve uncertain associations.
4. [Optional] Human operator intervention can be invoked by any analysis at anytime.
5. Deposit the associations into a linkbase through the XLink interface.

Now, a cursory correlation of the operational tasks that X-Map performs reveals the following design/implementation questions that must be answered. The motivation here is that

---

<sup>1</sup>Currently, this interface is either DOM2 or SAX2.

what X-Map *does* conceptually and how those concepts break down operationally is clear, but the design/implementation details of the operational tasks has not been tackled. The questions are:

1. What types of relations are valuable and/or feasible for X-Map? This classification directly affects the sorts of correlations that X-Map can make for operational task #3.
2. What are the algorithms and processes necessary to perform the above classification, and how should they connect together to discover the relations? The answer to this affects operational tasks #2, #3, and #4.
3. How does X-Map acquire and preserve the knowledge of the relations it just gleaned through its algorithms? This impacts operational tasks #1 and #5.

Furthermore, we consider the following three design goals influential on the X-Map design, and they will guide the implementation of the operational tasks. [34]

**Modularity:** The ability to extend X-Map's analytical capacity in discovering new relations and validating existing relations is critical for its forward evolution.

**Use of XML natively:** Using XML as natural inputs and outputs is important. Using XML as a mere persistence medium rather than a structured, hierarchical medium does *not* leverage the advantages of XML.

**Reuse of XML specifications:** The usage of existing/upcoming XML specifications and interfaces leverages not only the design expertise of the W3C but also the time and effort spent by the XML community in constructing, testing, and verifying the published public interfaces.

Thus, it behooves one to investigate in detail the questions that X-Map must answer in order to accomplish its operational tasks because they lie at the heart of XML Interoperability. In particular, X-Map's design aims to resolve data conflicts, including:

**Schematic Conflicts:** Data Type Conflict, Labelling Conflict, Aggregation Conflict, and Generalization Conflict

**Semantic Conflicts:** Naming Conflict and Scaling Conflict

As the reader may recall, these conflicts encompass a great majority of the interoperability problems introduced by design autonomy and data heterogeneity. Their impacts still exist for this project, but additional factors, including a wealth of relevant domain knowledge, help equalize the balance of power and enable X-Map to solve a suitably constrained case of the automatic association problem. The influence of these real but non-design factors on X-Map's design will also be looked at simultaneously in the following sections, along with their accompanying rationale.

Basically, X-Map employs a host of language, synonym, and structural analysis algorithms to meet and resolve the XML Interoperability problems posed by the project's requirements and goals. Armed with a regenerative association engine<sup>2</sup> [34] that constantly re-analyzes and refines associations when given new knowledge and aided by specific knowledge of the project's problem domain, X-Map can resolve the XML Interoperability problems.

A key feature of X-Map is the regenerative association engine (see Section 3.1.4 for a full description), which not only validates known associations but also speculates on uncertain associations. The regenerative association engine does this by keeping track of past associations that were denoted as not certain (i.e. associations where some participating

---

<sup>2</sup>Private Conversation: This insightful idea came from the experiences of Aditya P. Damle from his efforts at deriving and matching what Web documents mean with what people what. <http://www.2028.com/>

element's role was not determined) and trying to re-ascertain them when given information from newly acquired schemas. Thus, X-Map never really “gives up” on an association until it is proven to not exist.

The following sections will examine the three key questions enumerated on page 29 and give X-Map's answers and corresponding design. Namely, what relation classes do X-Map care about and why, what are the algorithms and processes used by X-Map to discover those relations, and how does it store the discovered relations?

### **3.1.1 Enumeration of Relation Classes**

#### **The Definition of Equivalence**

An important definition to nail down before discussing the merits of any type of relations is the meaning of the word *equivalence* — that is, under what circumstances do two objects become semantically and operationally indistinguishable from each other and therefore can be used interchangeably in each other's place?

This definition is important because the entire subject of interoperability revolves around this question. If one does not define, in some given context(s), what *equivalence* means, then interoperability is doomed from any mediator's perspective — there will be no common ground to even settle on to mediate between its client systems.

Furthermore, the structural analysis portion of X-Map's strategy depends on being able to identify and correlate parts of different schemas that are *equal* to each other. This is due to the schematic nature of XML documents and X-Map's strategy, which is to doubly-exploit structural and semantic information to achieve XML Interoperability (for details see Section 3.1.3). Without defining *equivalence*, X-Map cannot even exploit an XML

document's structure since it'd have no common elements with which to reason. Hence, X-Map's effectiveness relies heavily on this definition.

For this project, *equivalence* has the following meaning:

Two entities are equal when they express essentially the same physical object or concept and differ only by measurement units, mathematical precision, acronyms and/or abbreviations, or other shortening or lengthening of its labelling identifier by concatenation.

### **The Implications of Equivalence**

For example, X-Map will consider an element that measures time in milliseconds as equal to an element that measures time in seconds since they differ by a scalar factor. Likewise, X-Map will consider SSN and Social Security Number to be equal since the former is an acronym of the latter.

However, in the case of precision, one must pay careful attention when a relation reduces precision because of the semantic problems due to inappropriate numeric truncation. For example, suppose a number has range  $[0.0, 10.0]$  in domain 1 while its counterpart in domain 2 has range  $[0, 10]$ . While there is no obvious danger transforming from domain 1 to domain 2 (transforming a 6.6 to 7 is safe since domain 2 does not care about the precision), the reverse is not necessarily safe. Namely, a 7 in domain 2 could map to  $[6.5, 7.5)$  in domain 1, and a naive transformation to 7.0 could introduce a whopping 50% average error.

The thorny issue of precision and truncation represents a large class of interoperability problems stemming from information-loss, which unfortunately is not included in the scope

of this thesis and represents a future area of research. However, X-Map can help alleviate this problem by noting associations which involve loss of precision, flagging them programmatically, and saving this knowledge along with the association into persistent storage to serve as warnings to other applications.

The astute reader should realize that this definition of *equivalence* immediately addresses two of the semantic conflicts by making their resolution a part of X-Map: naming and scaling.

The naming conflict (i.e. SSN vs. S.S.N. vs. Social Security Number), by definition, is declared a trivial issue — X-Map only needs to detect the presence of this case and if so, determine if any element in the set of known abbreviations and acronyms of one element intersect with the same of the other, and then declare the elements equal if the intersection is non-empty. This tactic applies due to the fact that the set formed by the “abbreviations and acronyms” relation for any given element is a connected component<sup>3</sup> — every element in the set is related through some transitive set of equivalence relations to every other element in the set. Thus, if one element of a connected component is postulated to belong to two different components (i.e. the components of two elements in different data-domains), then the components must be the same; thus, the elements must be the same.

Likewise, the scaling conflict, such as English vs. Metric, is a vacuous issue — X-Map simply needs to detect if the elements involved are even related to measurement scales that it knows and cares about and if so, declare them equal and generate the appropriate mathematical transformation between the two measurement scales. The hard part here is the generation of the appropriate translation formula, but on-going work at MITRE exists to tackle this problem and can be sufficiently leveraged.

---

<sup>3</sup>A *connected* component refers to the concept in topology which says “if a component is connected, then a path (of relations) lead from every element in the component to every other element in the component”.

For this project, these SI [64] measurement units hold interest:

- Time — from seconds to microseconds
- Mass — in kilograms or pounds
- Length — in kilometers or miles
- Electro-magnetic radiation — in Watts or Joules
- Location — in longitude/latitude/degree/minute/second/decisecond

Considering the restricted nature of abbreviations/acronyms and scaling and the well-defined solutions to their behaviors, one can easily use a solution similar to that proposed by COIN to attack this problem — a lookup table of relevant values [6].

X-Map goes one step better with the well-known lookup table solution to this problem — its “lookup table” will actually be associations in the same linkbase of knowledge that it is building all this time. This provides two obvious benefits by re-leveraging the regenerative association engine:

1. Every use of a known association validates it.
2. Every introduction of new information into the linkbase of knowledge is an opportunity to resolve an unknown association.

### **Relation Classes based on Equivalence**

Finally, using the above definition of *equivalence* as the fundamental building block, we can describe the other relevant classes of relations for the project. They were summarized earlier in Table 1.1.

In line with the earlier assertion of how X-Map combats information-loss (see Section 3.1.1), X-Map recognizes an associated attribute list for each relation denoting whether the relation loses information upon traversal and if so, what type of lossage. Precision represents one possible value; other hints can be added but remains unspecified for this project.

X-Map also recognizes a similar attribute list for information-gain. One possible value is invertibility, which refers to whether the relation from one domain to the other can be reused in the opposite direction. Invertible mathematical formulas (Abstraction) and LookupTable obviously fall under this category.

### **3.1.2 Discovering Relations**

The previous section focused on identifying the relations that interest X-Map. This section will focus on how to identify those relations in XML documents, as well as some background preparation, which will then pave way for the discussion of algorithmic strategies in Section 3.1.3 that discovers these relations.

Like COIN’s axiomatic approach toward mediating queries [6], the following set defines X-Map’s operational “axioms”. This approach is feasible due to its emphasis on document structure, which directly leverages specific domain knowledge and will be discussed in Section 3.1.3.

#### **Algorithm Execution Decision**

Basically, X-Map performs an optimization step between loading the XML schema into memory and processing it with its algorithms — X-Map pre-processes the schema and

tries to compile a processing strategy on the fly, which X-Map then executes.

X-Map presently employs a simple heuristic to determine which of its algorithms to run on input XML schema. In this case, X-Map uses a computationally cheap circularity check, which, as we will shortly discuss in Section 3.1.3, is a good indicator of aggregation conflicts. Other cheap heuristics can be performed at this stage, each relevant to the algorithm it precedes, which will ultimately drive down computational costs while increasing relevant matchings.

However, this is purely an exercise in optimization to finding the right set of algorithms to run, not interoperability, so X-Map may, for simplicity, run all of its heuristics such as the circularity check.

## **Structural Analysis Background**

The major realization underlying X-Map’s structural analysis approach is the formulation of the data structure. This is important because it determines the types and number of analyses that can be run.

Due to the highly hierarchical and tree-like structure of XML schemas, the importance to recast this tree into a more computationally palatable form cannot be underestimated. The fundamental problem with structural analysis on a tree is the bias it builds against correlating aggregation and generalization conflicts. In non-trivial cases, these conflicts typically result in completely different tree structures which confound any attempts to frame any resolutions based on hierarchies. Thus, for XML Interoperability purposes, the schema’s “tree” must be altered into a better representation.

This realization leads X-Map to propose a non-traditional view of a tree — as a directed

graph — for XML Interoperability, as shown in the next chapter in Figure 4-2. Basically, tree nodes map to graph vertices, tree edges map to graph edges, and the edge direction goes from the root node to the child node.

This view transformation dramatically increases the number of interesting properties (of graphs) and also the analysis options one can perform on a tree hierarchy, and as Section 3.1.3 will show, this transformation can be done quickly.

Thus, the following list represents some<sup>4</sup> of the interesting features of a directed graph which X-Map will take advantage of, and they will be discussed in Section 3.1.3. Basically, each represents an embodiment of schematic or semantic conflicts mentioned earlier, and shows how X-Map uses the information.

- Associations in Hierarchies
- Associations in Cycles
- Associations through Contradiction

### **3.1.3 X-Map's Strategies**

As briefly mentioned in Section 3.1, X-Map's strategy employs a number of analysis algorithms on structure and language, aided with specific knowledge of the project's problem domain, to meet and resolve the XML Interoperability problem.

A key feature of X-Map is its regenerative association engine (see Section 3.1.4), which not only validates known associations but also keeps track of uncertain associations so that they can be ascertained when new future information becomes available to X-Map.

---

<sup>4</sup>By no means is this list all inclusive. Additional graph features to exploit can easily be the topic of future research.

Equally critical to X-Map's strategy is the idea of recasting a hierarchical data structure like XML into a directed graph and leveraging the body of graph theory work on it.

Finally, at all times, human intervention may prove more fruitful in resolving the final outstanding issues after each sub-analysis has finished weeding through the complexity. This proves to be a benefit to both the operator and the algorithm since the algorithm is ultimately not sentient and may require suggestions only a human can make, so the algorithm will go through the complexity which overwhelms most humans to extract the basic conflict in question.

Thus, the following sections will explain the process and rationale behind the graph analyses that X-Map uses along with how the regenerative association engine "does its magic".

### **How to Collapse a Tree Hierarchy**

The process of converting XML into a directed graph and vice versa proves deceptively simple due to existing software for another project at MITRE. The basic concept behind the "Bags" concept is that a hierarchy is simply a collection of nodes whose edges represent some attribute or containment relationship between the appropriate parent and child node. [34]

However, instead of focusing on how the relations stack together to form a tree-like hierarchy, if one focuses on the nodes and the relations they contain or participate, the graph formulation immediately leaps forth. Thus, "XML-2-Bags" embodies this realization to fruition. "XML-2-Bags", in pseudo-code, simply:

Also, one must realize that in addition to "flattening" the tree into a directed graph, X-Map will keep the hierarchical information around to aid its graph algorithms in determining

relations.

Finally, as explained earlier in Section 3.1.2, viewing a tree as a directed graph holds much potential, as the following sections will show.

### **Discovering Associations in Hierarchies**

Discovering associations between elements in a hierarchy can be tricky because the hierarchy is not guaranteed to contain any “meaning” for the associated elements. Fortunately, in this project, it is often the case that hierarchy embeds information about its constituent elements. This is domain knowledge specific to this project that will be exploited.

The typical example (with A, B, C, D, E as elements and arrows pointing in the to part of the relation) would be: A contains B; B contains C; and D contains E. If C and E are related, does that say anything about B and D? What about A and D?

For this project, it often turns out that if C and E are related, then either A or B are related to D. However, even if the relation cannot be immediately drawn from A or B to D, one can still speculate on its existence. Future processes may discover that B and D are related without the presence of C and E, in which case X-Map now has a more complete picture than either processes alone.

Furthermore, if a correlation is drawn both between C and E and an encompassing parent such as A and D, that more than likely sheds more light about the correlation of elements in between. Perhaps the intervening elements in one schema correspond to additional and yet discovered detail for the other schema.

## **Deriving Associations in Cycles**

Other times, an association can be made that is cyclic — that is, there is an association between an encompassing element and the contained element between the schemas. An example of this would be: A contains B; C contains D. A is related to D and C is related to B.

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 this conflict shows the advantages of the graph over the tree — a tree will steadfastly maintain such hierarchical conflicts during analysis and frustrate its resolution while a graph does not.

As noted in Section 3.1.3, this situation potentially allows X-Map to speculate on the meaning of intervening elements, too. Perhaps element E represents details from schema 1 that schema 2 does not yet exhibit and can be confirmed in the future through speculation.

## **Dealing with Conflicting Associations**

Clearly, it is possible to derive contradictory associations such as “A implies B and A does not imply B” or one-sided associations such as “A implies B but B does not imply A”. The latter situation possibly implies a substitutable relation while the former is slightly thornier. In general, situations like this can be handled by having a persistent body of knowledge that either affirmatively says “yes, A implies B”, or “no, A does not imply B”. The fully qualified means of building this knowledge is an active area of research.

X-Map builds this body of knowledge through its relations linkbase in a couple of ways.

- Some schemas may be more trusted than others; thus, relations drawn from them may have higher weight to break contradictory associations.
- X-Map can fully speculate on this result and wait for future schemas to resolve this issue via the regenerative association engine.

### 3.1.4 Regenerative Association Engine

The regenerative association engine embodies a simple logical concept extremely applicable toward XML Interoperability:

One should *never* throw away information, even uncertain ones, when trying to reason out and associate things — one never knows when one will need that information again.

Hence, The engine's design is surprisingly simple since its tasks basically entail the following:

1. Keep track of the speculative relations and what association algorithm produced each.
2. Rerun the associated algorithm on the speculative relations when new schemas are introduced into X-Map.
3. Record whether a speculative relation's conflict is resolved affirmatively or negatively and act accordingly.

## 3.2 Heuristics

Discovering associations between elements in a hierarchy can be tricky because the hierarchy is not guaranteed to contain any “meaning” for associated elements. Fortunately, in this project, it is typically the case that hierarchy embeds information about its constituent elements. This is domain knowledge specific to this project that will be exploited. In the case of heuristics, we use fairly simple 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 or Data Analysis on the domains in question, or reading off a set of known associations from a Linkbase.

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  $E$ , we can speculate with some (initially low) degree of confidence that  $C$  and  $F$  are related. We also acquire (slightly higher, but still low) confidence that  $A$  and  $D$  are related, or their parents.

The next step is noting that the converse is somewhat stronger: If *both*  $A - D$  and  $C - F$  are related, we have 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 our confidence in the association currently being examined. If the known associations are fairly distant, however, our level of confidence in these two elements being related drops off rapidly.

The score-combining function from the previous section comes in 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 more such known associations lie in the vicinity of the speculated edge, the more the edge score gets reinforced. We score nodes 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, if there was at some stage ambiguity 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: we have this flexibility in how their relationship is affected.

Other heuristics which could be used include deriving associations in cycles (this would necessitate 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 to us such associations that we might not have otherwise detected. Finally, we can often usefully “chain” associations. 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 the intermediate associations are not definite but speculative, the idea becomes cloudier, but we can still use probability-combination type formulae such as described in the preceding section. What makes this possible is our mode of storing multiple data domains in a single graph representation.

### 3.3 Graph Theory Analysis

The base problem we're dealing with — finding equivalencies in (unlabeled) graphs — has been fairly deeply analyzed computationally. For certain classes of graphs, solving graph isomorphism is provably NP-complete. There are known randomized algorithms for graph isomorphism, however, (mostly due to Babai and Erdos) that run in  $O(n^2)$  time on “almost

all” pairs of graphs, i.e., pathological instances for which these algorithms fail are highly unlikely to turn up in real-world situations. The algorithms still need to be tweaked to apply to our problem, for the following reasons:

1. We aren't trying to determine perfect isomorphism, just "likely" links.
2. During analysis we'll already have some of the nodes 'labeled' in the sense that we'll already know how some of them correspond to each other.
3. What makes the problem much harder, however, is the fact that the graphs in our problems are not even perfectly isomorphic, but merely "similar". There will always be some nodes with no correspondencies in the other domain, as well as nodes for which the hierarchy, parenthood structure is not matched by its analogue on the other side.

A logical way to try to attack the problem, aided by definite knowledge of some associations, is to guess the identity of some  $k$  vertices in one domain with  $k$  in the other, and see what the assumption of these implies. Depending on the initial assignment of  $k$  nodes, if we look at powers of the adjacency matrix whose rows are the nodes we have fixed, they give a classification of the columns that will break them up enough to distinguish them. or to make the ones possibly similar to one another much smaller than before.

How efficient will this procedure be? In the worst case, we have to worry about graphs that look like projective planes, for which it would be prohibitively unlikely to choose the correct  $k$  nodes in a reasonable number of tries. in the practical case we can possibly use a probabilistic argument averaging over the class of graph problems we derive from XML trees. There is some subjectivity in describing our general data structures in this numerical fashion, but it should not be excessive.

## 3.4 Other Techniques

My thesis finally focuses on other potential techniques to use for structural analysis, drawn from artificial intelligence. The use of continuous scores opens up a vast number of mathematical operations that can be performed on sets of edges. When we bring into play feedback loops that repeatedly update the scores of edges over several cycles, we make possible the application of neural nets, Bayesian analysis and other techniques which will also take into account existing data and requisite domain knowledge. Essentially, the thesis will provide a set of comparisons of these various analyses to determine which ones work best in the context of the project.



# Chapter 4

## 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 X-Map Linkbase [34]. Heuristics, like most of the techniques used in structural analysis, serve a complementary rather than a stand-alone role. Figure 4-1 [34] shows an example of where heuristic analysis would fall into the larger picture.

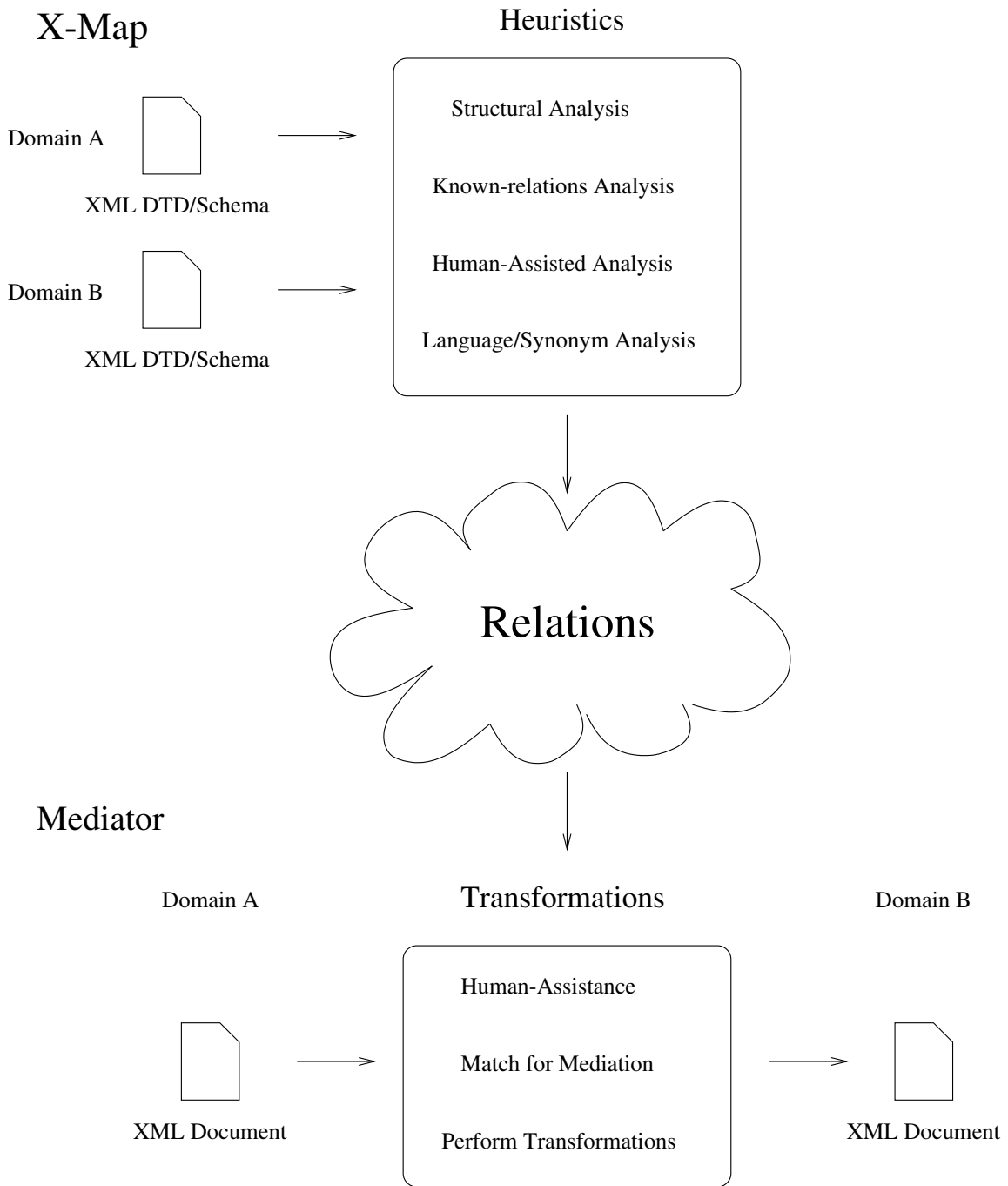


Figure 4-1: Heuristics Application

## 4.1 Reasoning from Associations

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.

## 4.2 Associations in Cycles

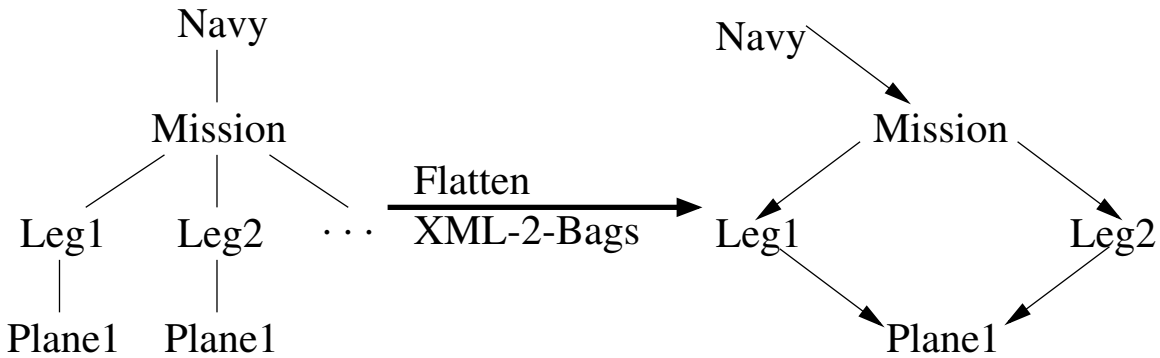


Figure 4-2: Deriving Associations in Cycles

Other methods used include deriving associations in cycles, as in Figure 4-2 [34]. 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.

## 4.3 Chaining Associations across Domains

The concept of “chaining” associations 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.



# Chapter 5

## Discovering Mappings using Graph Theory

### 5.1 Relation to Graph Problems

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 [45], 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 [52, 47]. Subgraph isomorphism, i.e. finding the largest isomorphic subgraphs of given graphs, is also a famous NP-complete problem [46]. There are known randomized algorithms [44, 43] 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  $B$  [45]. 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; [45] gives a reduction to the well-known MAX-CLIQUE problem.

## 5.2 Solving Isomorphism via Adjacency Matrices

Approaches proposed by other researchers to attack this problem include NAUTY [48], the Graph Matching Toolkit [49], and Combinatorica [50]. 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.

## 5.3 Other Avenues

Although not used in the prototype, one alternative strategy was analyzed in detail. Erdős and Gallai's classic extremal function [53] 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 [29]. 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 [54].

We can prove an exact lower bound on  $\gamma(G)$ , the size of the smallest matching that a certain  $O(m+n)$  time greedy matching procedure may find for a given graph  $G$  with  $n$  vertices and  $m$  edges. The bound is precisely Erdős and Gallai's extremal function that gives the size of the smallest maximum matching, over all graphs with  $n$  vertices and  $m$  edges. Thus the greedy procedure is optimal in the sense that when only  $n$  and  $m$  are specified, no algorithm can be guaranteed to find a larger matching than the greedy procedure. The greedy procedure and augmenting path algorithms are seen to be complementary: 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 are shown to be faster than the augmenting path algorithm alone. The lower bound on  $\gamma(G)$  is a stronger version of Erdős and Gallai's result, and so the proof of the lower bound is a new way of proving of Erdős and Gallai's result.

The following procedure is sometimes recommended for finding a matching that is used as an initial matching by a maximum cardinality matching algorithm [29]. Start with the empty matching, and 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, and remove  $v$  and  $w$  from the graph. This procedure is referred to in this paper as “the greedy matching procedure” or “the greedy procedure.”

In the worst case, the greedy procedure performs poorly. For all  $r \geq 3$ , a graph  $D_r$  of order  $4r + 6$  can be constructed such that the greedy procedure finds a matching for  $D_r$  that is only about half the size of a maximum matching [32]. This performance is as poor as that of any procedure that finds a maximal matching.

On the other hand, there are classes of graphs for which the greedy procedure always finds a maximum matching [32]. Furthermore, using a straightforward kind of priority queue that has 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 [55]. The  $O(m + n)$  running time is asymptotically faster than the fastest known maximum matching algorithm for general graphs or bipartite graphs [21, 23, 24, 25, 26, 27, 28, 30]. The greedy procedure's success on some graphs,  $O(m + n)$  time and storage requirements, low overhead, and simplicity motivate the investigation of its performance.

The matching found by the greedy procedure may 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.

We will show that each graph  $G$  with  $n$  vertices and  $m \geq 1$  edges satisfies

$$\gamma(G) \geq \min\left(\left\lfloor n + \frac{1}{2} - \sqrt{n^2 - n - 2m + \frac{9}{4}} \right\rfloor, \left\lfloor \frac{3}{4} + \sqrt{\frac{m}{2} - \frac{7}{16}} \right\rfloor\right). \quad (5.1)$$

It will become clear that this bound is the best possible — when only  $n$  and  $m$  are given, no algorithm can be guaranteed to find a matching larger than that found by the greedy procedure.

The simpler but looser bound of  $\gamma(G) \geq m/n$  is proved in [55].

The bound in (5.1) can be considered 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  $\omega(m + n)$ . It is traditional to use a hybrid algorithm: first, use the greedy procedure (or one like it) to find a matching  $M$  in  $O(m + n)$  time; then, run an augmenting path algorithm with  $M$  as the initial matching. We will see that (5.1) supports the use of such hybrid algorithms. Intuitively, if the input graph is dense, then the greedy procedure finds a large matching, and the augmenting path algorithm needs only a few augmentation phases; if the input graph is sparse, then each augmentation phase is fast.

We can abstract the following technique for solving maximum cardinality matching problems: use one kind of method (perhaps the greedy procedure) for handling dense graphs, and another kind of method (perhaps an augmenting path algorithm) for handling other graphs. It may be interesting to investigate whether existing matching algorithms can be improved upon by explicitly using this technique.

## 5.4 Definitions and Notation

We consider the problem of finding a maximum matching in finite simple undirected unweighted possibly non-bipartite graphs.

Let  $G = (V, E)$  be a graph. We use  $vw$  as an abbreviation for an edge  $\{v, w\} \in E$ . For  $v \in V$ , the graph  $G - v$  is the graph with vertex set  $V - v$ , and edge set  $\{xy \in E : x \neq v \text{ and } y \neq v\}$ . The number of vertices and edges in  $G$  are respectively  $n(G)$  and  $m(G)$ . The degree of a minimum degree vertex of  $G$  is denoted  $\delta(G)$ . An edge  $vw \in E$ ,  $\deg v \leq \deg w$ , is called *semi-minimum* if  $\deg v = \delta(G)$  and  $\deg w$  is minimum over the degrees of  $v$ 's neighbors. 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  $K_n$ ; its complement is the *heap*  $\overline{K}_n$ .

Function arguments are sometimes omitted when the context is clear, e.g.,  $\nu$  may be used instead of  $\nu(G)$ . The notation  $a =^* b$  indicates that some algebraic manipulation showing that  $a = b$  has been omitted so as to shorten the presentation.

## 5.5 A Related Theorem

The analysis of the greedy matching procedure is closely related to the following theorem.

**Theorem 1 (Erdős and Gallai, 1959)** *The maximum number of edges in a simple graph*

of order  $n$  with a maximum matching of size  $k$  ( $2 \leq 2k \leq n$ ) is

$$\begin{cases} \binom{k}{2} + k(n-k) & \text{if } k < \frac{2n-3}{5}, \\ \binom{2k+1}{2} & \text{if } \frac{2n-3}{5} \leq k < \frac{n}{2}, \\ \binom{2k}{2} & \text{if } k = \frac{n}{2}. \end{cases}$$

Erdős and Gallai's theorem can be proved in one direction by considering three graphs: the graph obtained by connecting every vertex of  $K_k$  to every vertex of  $\overline{K}_{n-k}$ ; the graph  $K_{2k+1}$ ; and the graph  $K_n$ . The edge counts appearing in the theorem are the number of edges in these graphs. Thus the indicated edge counts can be realized for a given value of  $k$ .

Theorem 1 implies that if a graph has more than the indicated number of edges as a function of  $k-1$ , then the matching number of the graph is at least  $k$ . This fact, which is essentially equivalent to Theorem 1, is stated explicitly below.

**Corollary 2** *Let  $G$  be a graph with  $n$  vertices and  $m$  edges, and let  $k$  be an integer such that*

$$m \geq \begin{cases} \binom{k-1}{2} + (k-1)(n-k+1) + 1 & \text{if } k \leq \frac{2n+2}{5}, \\ \binom{2k-1}{2} + 1 & \text{if } k \geq \frac{2n+2}{5}. \end{cases}$$

*Then  $\nu(G) \geq k$ .*

(In Corollary 2, when  $k = \frac{2n+2}{5}$ , both conditions apply; they are equivalent.)

## 5.6 The Hybrid Approach

The  $O(m\sqrt{n})$  time general matching algorithms of Micali and Vazirani [28, 31] and Blum [23] operate in phases. Each phase uses  $O(m)$  time, and there are at most  $2\sqrt{\nu}$  phases. A matching  $M$  is maintained; initially,  $M$  has size, say,  $\alpha$ ,  $0 \leq \alpha \leq \nu$ . Each phase except the last enlarges  $M$ , so there are at most  $\nu - \alpha + 1$  phases. A bound on the running time  $T_g$  of these general matching algorithms, therefore, is

$$T_g = O(m \cdot \min(2\sqrt{\nu}, \nu - \alpha)). \quad (5.2)$$

(This bound and others in this section are actually too low by a  $O(m)$  term. For simplicity this is ignored in the remainder of this section.)

Now consider a hybrid algorithm that finds an initial matching in  $O(m + n)$  time using the greedy procedure, and then uses one of the  $O(m\sqrt{n})$  general matching algorithms. We have

$$\alpha \geq \gamma \geq \min\left(\left\lfloor n + 1/2 - \sqrt{n^2 - n - 2m + 9/4} \right\rfloor, \left\lfloor 3/4 + \sqrt{m/2 - 7/16} \right\rfloor\right). \quad (5.3)$$

Substituting into (5.2) yields a bound on the running time  $T_h$  of a hybrid algorithm:

$$T_h = O(m \cdot \min(2\sqrt{\nu}, \nu - \min(n - \sqrt{n^2 - n - 2m}, \sqrt{m/2}))). \quad (5.4)$$

This bound is tighter than  $O(m\sqrt{\nu})$  for graphs that are dense relative to  $\nu$ .

Let us see what happens when (5.4) is used to obtain a bound that is in terms of only  $n$  and  $m$ . Substituting  $\nu \leq n/2$  yields

$$T_h = O(m \cdot \min(2\sqrt{n/2}, n/2 - \min(n - \sqrt{n^2 - n - 2m}, \sqrt{m/2}))).$$

The  $n - \sqrt{n^2 - n - 2m}$  term turns out to be redundant; eliminating it gives

$$T_h = O(m \cdot \min(2\sqrt{n/2}, n/2 - \sqrt{m/2})). \quad (5.5)$$

The right side of (5.5) reduces to  $O(m\sqrt{n})$  unless  $m$  is  $\Theta(\binom{n}{2})$ ; thus (5.5) is almost no improvement over  $O(m\sqrt{n})$ . In practice, however, it might be useful to bound the number of phases by using the non-asymptotic version of (5.5).

The bounds (5.4) and (5.5) imply a complementary relationship between the greedy procedure and general matching algorithms that use repeated  $O(m)$  time augmentation phases. For dense graphs, the greedy procedure finds a large matching, and few augmentation phases are needed; for sparse graphs, each augmentation phase is fast. Although hybrid algorithms has long been considered to give better performance than, say, using Micali and Vazirani's algorithm alone [29], this specific complementary relationship seems not to have been generally known.

Since the  $O(m\sqrt{n})$  general matching algorithms are complicated [31], a less complicated but possibly slower algorithm is sometimes preferred. For example, one might do just one augmentation per phase [29]. This can require as many as  $n/2$  augmenting phases, as opposed to  $O(\sqrt{n})$  phases. In this case the greedy procedure's performance bounds take on a larger role. An analysis similar to the one earlier in this section shows that the running time for the resulting hybrid algorithm is

$$O(m \cdot \max(\sqrt{n^2 - n - 2m} - n/2, n/2 - \sqrt{m/2})). \quad (5.6)$$

For dense graphs this is a significant improvement over  $O(mn)$ .

All graphs considered here are finite, undirected and simple, unless otherwise noted. Let  $H = (V_H, E_H)$  be a graph. An *H-covering design* of a graph  $G = (V_G, E_G)$  is a set

$L = \{G_1, \dots, G_s\}$  of subgraphs of  $G$  such that each  $G_i$  is isomorphic to  $H$  and every edge  $e \in E_G$  appears in at least one member of  $L$ . The  $H$ -covering number of  $G$ , denoted by  $cov(G, H)$ , is the minimum number of members in an  $H$ -covering design of  $G$ . (If there is an edge of  $G$  which cannot be covered by a copy of  $H$ , we put  $cov(G, H) = \infty$ ). Clearly,  $cov(G, H) \geq |E_G|/|E_H|$ . In case equality holds, the  $H$ -covering design is called an  $H$ -decomposition (or  $H$ -design) of  $G$ . Two trivial necessary conditions for a decomposition are that  $|E_H|$  divides  $|E_G|$  and that  $gcd(H)$  divides  $gcd(G)$  where the  $gcd$  of a graph is the greatest common divisor of the degrees of all the vertices. In case  $G = K_n$ , the two necessary conditions are also sufficient, provided  $n \geq n_0(H)$ , where  $n_0(H)$  is a sufficiently large constant. If, however, the necessary conditions do not hold, the best one could hope for is an  $H$ -covering design of  $K_n$  where the following three properties hold:

1. **2-overlap:** Every edge is covered at most twice.
2. **1-intersection:** Any two copies of  $H$  intersect in at most one edge.
3. **Efficiency:**  $s|E_H| < \binom{n}{2} + c(H) \cdot n$ , where  $s$  is the number of members in the covering, and  $c(H)$  is some constant depending only on  $H$ .

Our main result is that  $H$ -covering designs of  $K_n$ , having these three properties, exist for every fixed graph  $H$ , and for all  $n \geq n_0(H)$ : Let  $H$  be a fixed graph. There exists  $n_0 = n_0(H)$  such that if  $n \geq n_0$ ,  $K_n$  has an  $H$ -covering design with the 2-overlap, 1-intersection, and efficiency properties. This  $i$ -intersection would give a likely set of associations between nodes of the corresponding graphs, and therefore a mapping between the respective elements in the domains.



# Chapter 6

## Artificial Intelligence Based Analysis — Neural Nets

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.

### 6.1 Neural Nets

Neural Networks (NNs) fit into the fourth layer of a top-down algorithm architecture (Figure 6-1). NNs learn from past performance to determine candidates for the current session [57]. In the case of NNs, a description of the subgraph around a potential edge serves as

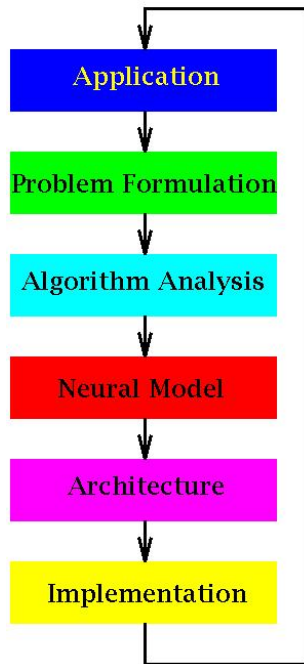


Figure 6-1: Algorithmic Approach

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 possess 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.

## 6.2 NNs in X-Map

Clifton and Li [56] 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.

The contribution of application-driven neural networks to structural analysis hinges upon three main characteristics:

1. Adaptiveness and self-organization: it offers robust and adaptive processing by adaptively learning and self-organizing.
2. Nonlinear network processing: it enhances the approximation, classification and noise-immunity capabilities.
3. Parallel processing: it employs a large number of processing cells enhanced by extensive interconnectivity.

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.

The neural model gets applied in two phases:

**Retrieving Phase** : The results are either computed in one shot or updated iteratively based on the retrieving dynamics equations. The final neuron values represent the desired output to be retrieved.

**Learning Phase** : The network learns by adaptively updating the synaptic weights that characterize the strength of the connections. Weights are updated according to the information extracted from new training patterns. Usually, the optimal weights are obtained by optimizing (minimizing or maximizing) certain "energy" functions. In our case, we use the popular least-squares-error criterion between the teacher value and the actual output value.

## 6.3 Implementation

The implementation was based on the toolkit JANNT (JAVa Neural Network Tool) [63] A supervised network was used. Figure 6-2 [62] schematically illustrates such a network, where the "teacher" is a human who validates the network scores.

Updating of weights in the various layers occurs according to the formula

$$w_{ij}^{(m+1)} = w_{ij}^{(m)} + \Delta w_{ij}^{(m)}$$

where  $w^{(m)}$  refers to the node weights at the  $m$ 'th iteration, and  $\Delta$  is the correcting factor. The training data consists of many pairs of input-output patterns.



For input vector  $x$ , the linear basis function  $u$  is the first-order basis function

$$u_i(w, x) = \sum_{j=1}^n w_{ij}x_j$$

and the activation function  $f$  is the sigmoid

$$f(u_i) = \frac{1}{1 + e^{-u_i/\sigma}}$$

The approximation based algorithm can be viewed as an approximation regression for the data set corresponding to real associations. The training data are given in input/teacher pairs, denoted as  $[\mathcal{X}, \mathcal{T}] = \{(x_1, t_1), (x_2, t_2) \dots (x_M, t_M)\}$ , where  $M$  is the number of training pairs, and the desired values at the output nodes corresponding to the input  $x^{(m)}$  patterns are assigned as teacher's values. The objective of the network training is to find the optimal weights to minimize the error between "target" values and the actual response. The criterion is the minimum-squares error observed.

The model function is a function of inputs and weights:  $y = \phi(x, w)$ , returning edge probability  $y$ . The weight vector  $w$  is trained by minimizing the energy function along the gradient descent direction:

$$\Delta w \propto -\frac{dE(x, w)}{dw} = (t - \phi(x, w)) \frac{d\phi(x, w)}{dw}$$

For X-Map, a standard feed-forward back-propagation network was used, with two layers, but further efforts could focus on other configurations as well.

# Chapter 7

## 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).

### 7.1 Technique Performances

Comparative results are shown in Table 7.1.

|                              | # actual mappings nominated | # non-mappings 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 7.1: Performance of Structural Analysis Techniques

Table 7.1 was constructed as follows. Two similar sample data domains were presented to the structural analysis algorithm. The domains were initially compared by a human interpreter, and mappings noted. Then, some of the obvious mappings were bootstrapped into the graph, to simulate prior knowledge discovered by equivalence analysis and data analysis. Clearly, these static analyses could not discover all relevant associations. The algorithm's task was to find previously undiscovered associations.

The score  $S_i$  of the  $i$ 'th technique was computed according to the proportion of associations it got right, minus its proportion of failures:

$$S_i = \frac{c_i}{T - K} - \frac{w_i}{F};$$

- $c_i$  and  $w_i$  are the number of correct and incorrect mappings, respectively, nominated

by the algorithm.

- $T$  is the total number of correct mappings that existed between the sampled domains,  $F$  is the number of “false” mappings (i.e. element pairs that had no relationship), and  $K$  is the number of mappings that were known ahead of time (i.e., discovered by some other analysis).

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.

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.

## **7.2 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 X-Map 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.

# **Appendix A**

## **Source Code**



# Bibliography

- [1] P.V. Biron, A. Malhotra, editors. XML Schema Part 2: Datatypes. W3C Working Draft, April 7, 2000. <http://www.w3.org/TR/xmlschema-2>
  
- [2] T. Bray, J. Paoli, C.M. Sperberg-McQueen, editors. eXtensible Markup Language (XML) 1.0. W3C Recommendation, February 10, 1998. <http://www.w3.org/TR/REC-xml>
  
- [3] J. Clark, S. DeRose, editors. XML Path Language (XPath) Version 1.0. W3C Recommendation, November 16, 1999. <http://www.w3.org/TR/xpath>
  
- [4] D.C. Fallside, editor. XML Schema Part 0: Primer. W3C Working Draft, April 7, 2000. <http://www.w3.org/TR/xmlschema-0>
  
- [5] Ashish Mishra, Michael Ripley, Amar Gupta. Using Structural Analysis to Mediate XML Semantic Interoperability. MIT Sloan Working Paper No. 4345-02, February 2002. [http://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=306845](http://papers.ssrn.com/sol3/papers.cfm?abstract_id=306845)
  
- [6] C.H. Goh. Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems. PhD dissertation. Massachusetts Institute of Technology, Sloan School of Management, December 1996.

- [7] C.F. Goldfarb, W.E. Kimber, P.J. Newcomb, S.R. Newcomb, editors. ISO/IEC 10744:1997 Architectural Form. August 27, 1997. <http://www.ornl.gov/sgml/wg8/docs/n1920/html/toc.html>
- [8] W. Kim and J. Seo (1991). Classifying schematic and data heterogeneity in multi-database systems. *IEEE Computer*, 24(12):12-18.
- [9] R. Krishnamurthy, W. Litwin, and W. Kent (1991). Language features for interoperability of databases with schematic discrepancies. In *Proceedings of the ACM SIGMOD Conference*, pages 40-49.
- [10] D. Megginson. *Structuring XML Documents*. 1998. Prentice Hall.
- [11] C.F. Naiman and A.M. Ouskel (1995). A classification of semantic conflicts in heterogeneous database systems. *Journal of Organizational Computing*, 5(2):167-193.
- [12] Report on Information Management to Support the Warrior, SAB-TR-98-02. December 1998. <http://afosr.sciencewise.com/afr/sab/any/text/any/afrtstud.htm>
- [13] <http://diides.ncr.disa.mil/shade>
- [14] D. Brickley, R.V. Guha, editors. Resource Description Framework (RDF) Schema Specification 1.0. W3C Candidate Recommendation, March 27, 2000. <http://www.w3.org/TR/rdf-schema>
- [15] H.S. Thompson, D. Beech, M. Maloney, N. Mendelsohn, editors. XML Schema Part 1: Structures. W3C Working Draft, April 7, 2000. <http://www.w3.org/TR/xmlschema-1>
- [16] J. Clark, editor. eXtensible Stylesheet Language Transform (XSLT) Version 1.0. W3C Recommendation, November 16, 1999. <http://www.w3.org/TR/xslt>
- [17] M. Biezunski, M. Bryan, S. Newcomb, editors. ISO/IEC FCD 13250:1999 Topic Map. April 19, 1999. <http://www.ornl.gov/sgml/sc34/document/0058.htm>

- [18] Y.J. Breitbart and L.R. Tieman (1985). ADDS: Heterogeneous distributed database system. In F. Schreiber and W. Litwin, editors, *Distributed Data Sharing Systems*, pages 7-24. North Holland Publishing Co.
- [19] P. Ion and R. Miner, editors. *Mathematical Markup Language*. W3C Proposed Recommendation. February 24, 1998. <http://www.w3.org/TR/PR-math/>
- [20] A. Sheth and V. Kashyap (1992). So far (schematically) yet so near (semantically). In D.K. Hsiao, E.J. Neuhold, R. Sacks-Davis. *Proceedings of the IFIP WG2.6 Database Semantics Conference on Interoperable Database Systems (DS-5)*, pages 283-312, Lorne, Victoria, Australis. North-Holland.
- [21] 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.
- [22] Claude Berge. *Graphs and Hypergraphs*. North-Holland Publishing Company, 1973.
- [23] 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.
- [24] 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.
- [25] Zvi Galil. Efficient algorithms for finding maximum matchings in graphs. *Computing Surveys*, 18:23–38, 1986.
- [26] 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.

- [27] L. Lovász and M. D. Plummer. *Matching Theory*, volume 121 of *North-Holland mathematics studies*. North-Holland, Amsterdam, 1986.
- [28] 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.
- [29] B. M. E. Moret and H. D. Shapiro. *Algorithms from P to NP*. The Benjamin/Cummings Publishing Company, Inc., 1991.
- [30] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice Hall, 1982.
- [31] Paul A. Peterson and Michael C. Loui. The general maximum matching algorithm of Micali and Vazirani. *Algorithmica*, 3:511–533, 1988.
- [32] 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.
- [33] 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.
- [34] D. Wang. Automated semantic correlation between multiple schema for information exchange. Master’s thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, 2000.
- [35] 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.
- [36] JBI — Joint Battlespace Infosphere. <http://www.rl.af.mil/programs/jbi/default.cfm>, August 2001.

- [37] Joint Battlespace Infosphere (JBI) Fuselet Research and Development. <http://www.rl.af.mil/div/IFK/prda/prda0009.html>, January 2002.
- [38] Protégé-2000. <http://protege.stanford.edu/index.shtml>, June 2001.
- [39] The COntext INterchange Project <http://context.mit.edu/~coin/>, July 2000.
- [40] C.F. Goldfarb, W.E. Kimber, P.J. Newcomb, S.R. Newcomb, editors. ISO/IEC 10744:1997 Architectural Form. <http://www.ornl.gov/sgml/wg8/docs/n1920/html/toc.html>, August 27, 1997.
- [41] M. Biezunski, M. Bryan, S. Newcomb, editors. ISO/IEC FCD 13250:1999 Topic Map. <http://www.ornl.gov/sgml/sc34/document/0058.htm>, April 19, 1999.
- [42] XML Linking Language (XLink) Version 1.0 <http://www.w3.org/TR/xlink/>, 27 June 2001
- [43] 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.
- [44] 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.
- [45] 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.
- [46] 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.

- [47] 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.
- [48] B. D. McKay. Nauty. <http://cs.anu.edu.au/people/bdm/nauty/>, February 2000.
- [49] GMT - Graph Matching Toolkit [http://www.cs.sunysb.edu/algorithm/implement/gmt/](http://www.cs.sunysb.edu/algorithm/implement/gmt/implement.shtml) [implement.shtml](http://www.cs.sunysb.edu/algorithm/implement/gmt/implement.shtml), July 1999
- [50] S. S. Skiena. *Implementing Discrete Mathematics: Combinatorics and Graph Theory in Mathematica*. Addison-Wesley, Redwood City CA, June 1990.
- [51] D. Kleitman, Personal Communication. MIT Mathematics Department (Combinatorics), June 2000.
- [52] K. Zhang and T. Jiang. Some MAX SNP-hard results concerning unordered labeled trees. *Inform. Process. Lett.* 49, 249-254, June 1994.
- [53] Paul Erdős and T. Gallai. On maximal paths and circuits of graphs. *Acta Math. Acad. Sc. Hungar.*, 10:337–356, 1959.
- [54] 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)
- [55] Andrew Shapira. Matchings, degrees, and  $O(m+n)$  time procedures, in preparation.
- [56] Wen-Syan Li and Chri Clifton. Semantic Integration in Heterogenous Databases using Neural Networks Proceedings of the 20th VLDB Conference, 1994
- [57] Patrick Henry Winston Artificial Intelligence Addison-Wesley Co., 1992
- [58] C. M. Sperberg-McQueen and Henry Thompson. XML Schema. W3C Architecture Domain, April 2000. <http://www.w3.org/XML/Schema>

- [59] 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>
- [60] P.V. Biron, A. Malhotra, editors. XML Schema Part 2: Datatypes. W3C Recommendation, May 2, 2001. <http://www.w3.org/TR/xmlschema-2>
- [61] J. Clark, editor. eXtensible Stylesheet Language Transform (XSLT) Version 1.0. W3C Recommendation, November 16, 1999. <http://www.w3.org/TR/xslt>
- [62] Francisco Rodriguez. Supervised and Unsupervised Neural Networks. November, 2001. <http://www.gc.ssr.upm.es/inves/neural/ann1/concepts/Suunsupm.htm>
- [63] Tom Doris. JAVa Neural Network Tool. July, 1999. <http://www.compapp.dcu.ie/~tdoris/BCK/>
- [64] NIST. International System of Units. July 1999. <http://physics.nist.gov/cuu/Units/index.html>